Broadview®
www.broadview.com.cn 博文视点



实战Nginx

取代Apache的高性能Web服务器

张宴 著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

VISIT…

LANZAROTE Caliente.com

目　录

第 1 章 Nginx 简介基本介绍
1.1　常用的 Web 服务器简介　3
1.1.1　Apache 服务器　3
1.1.2　Lighttpd 服务器　3
1.1.3　Tomcat 服务器　4
1.1.4　IBM WebSphere 服务器　4
1.1.5　Microsoft IIS　4
1.2　Nginx 简介　4
1.3　选择 Nginx 的理由　5
1.3.1　它可以高并发连接　5
1.3.2　内存消耗少　7
1.3.3　配置文件非常简单　8
1.3.4　成本低廉　8
1.3.5　支持 Rewrite 重写规则　9
1.3.6　内置的健康检查功能　9
1.3.7　节省带宽　9
1.3.8　稳定性高　9
1.3.9　支持热部署　9
1.4　Nginx 与 Apache、Lighttpd 的综合对比　9

第 2 章　Nginx 服务器的安装与配置　11
2.1　安装 Nginx 服务器所需要的系统资源　11
2.2　Nginx 的下载　12
2.3　Nginx 的安装　12
2.3.1　Nginx 在 Windows 环境下的安装　13
2.3.2　Nginx 在 Linux 环境下的安装　13
2.4　Nginx 的启动、停止、平滑重启　16
2.4.1　Nginx 的启动　17
2.4.2　Nginx 的停止　17
2.5　Nginx 的平滑重启　18
2.6　Nginx 的信号控制　18
2.7　Nginx 的平滑升级　19

第 3 章　Nginx 的基本配置与优化　21
3.1　Nginx 的完整配置示例　21
3.2　Nginx 的虚拟主机配置　23
3.2.1　什么是虚拟主机　23
3.2.2　配置基于 IP 的虚拟主机　24
3.2.3　配置基于域名的虚拟主机　27
3.3　Nginx 的日志文件配置与切割　29
3.3.1　用 log_format 指令设置日志格式　29
3.3.2　用 access_log 指令指定日志文件存放路径　30
3.3.3　Nginx 日志文件的切割　32
3.4　Nginx 的压缩输出配置　33
3.5　Nginx 的自动列目录配置　35
3.6　Nginx 的浏览器本地缓存设置　36

第 4 章　Nginx 与 PHP（FastCGI）的安装、配置与优化　38
4.1　获取相关开源程序　39
4.2　安装 PHP 5.2.10（FastCGI 模式）　42
4.3　安装 Nginx 0.8.15　52
4.4　配置开机自动启动 Nginx + PHP　55
4.5　优化 Linux 内核参数　55
4.6　在不停止 Nginx 服务的情况下平滑变更 Nginx 配置　56
4.7　编写每天定时切割 Nginx 日志的脚本　56

第 5 章　Nginx 与 JSP、ASP.NET、Perl 的安装与配置　59
5.1　Nginx 与 JSP（Tomcat）在 Linux 上的安装、配置　59
5.2.1　Tomcat 和 JDK 的安装　60
5.1.2　Nginx 与 Tomcat 的配置　61
5.2　Nginx 与 ASP.NET（Mono+FastCGI）在 Linux 上的安装、配置　63
5.2.1　Mono 的安装　63
5.2.2　Nginx 与 ASP.NET（Mono+FastCGI）的配置　64
5.3　Nginx 与 Perl（FastCGI）在 Linux 上的安装、配置　67
5.3.1　Perl（FastCGI）的安装　67
5.3.2　Nginx 与 Perl（FastCGI）的配置　70

第 6 章　Nginx HTTP 负载均衡和反向代理的配置与优化　73
6.1　什么是负载均衡和反向代理　73
6.1.1　负载均衡　73
6.1.2　反向代理　73
6.2　常见的 Web 负载均衡方法　74
6.2.1　用户手动选择方式　74
6.2.2　DNS 轮询方式　75
6.2.3　四/七层负载均衡设备　77
6.2.4　多线多地区智能 DNS 解析与混合负载均衡方式　81
6.3　Nginx 负载均衡与反向代理的配置实例　83
6.3.1　完整的 Nginx 反向代理示例如代码 6-3 所示　83
6.3.2　Nginx 负载均衡与反向代理实现动、静态网页分离　86
6.4　Nginx 负载均衡的 HTTP Upstream 模块　88
6.4.1　ip_hash 指令　89
6.4.2　server 指令　89
6.4.3　upstream 指令　90
6.4.4　upstream 相关变量　90
6.5　Nginx 负载均衡服务器的双机高可用　91

第 7 章　Nginx 的 Rewrite 规则编写实例　99
7.1　什么是 Nginx 的 Rewrite 规则　99
7.2　Nginx Rewrite 规则相关指令　99
7.2.1　break 指令　100
7.2.2　if 指令　100
7.2.3　return 指令　101
7.2.4　rewrite 指令　104
7.2.5　set 指令　106
7.2.6　uninitialized_variable_warn 指令　106
7.2.7　Nginx Rewrite 可以用到的全局变量　106
7.3　PCRE 正则表达式语法　107
7.4　Nginx 的 Rewrite 规则编写实例　109
7.5　Nginx 与 Apache 的 Rewrite 规则实例对比　112
7.5.1　简单的 Nginx 与 Apache Rewrite 重写规则　112
7.5.2　允许指定的域名访问本站，其他域名一律跳转到

http://www.aaa.com　113
7.5.3　URL 重写与反向代理同时进行　114
7.5.4　指定 URL 之外的 URL 进行 Rewrite 跳转　114
7.5.5　域名前缀作为重写规则变量的示例　115

第 8 章　Nginx 模块开发　117
8.1　Nginx 模块概述　117
8.2　Nginx 模块编写实践　119
8.2.1　Hello World 模块编写与安装　119
8.2.2　Hello World 模块分析　121

第 9 章　Nginx 的 Web 缓存服务与新浪网的开源 NCACHE 模块　127
9.1　什么是 Web 缓存？　127
9.2　Nginx 的 Web 缓存服务　128
9.2.1　proxy_cache 相关指令集　128
9.2.2　proxy_cache 完整示例　130
9.2.3　fastcgi_cache 相关指令集　131
9.3　新浪网开源软件项目——基于 Nginx 的 NCache 网页缓存系统　135
9.3.1　NCACHE 模块的安装　136
9.3.2　NCACHE 配置文件编写　136
9.3.3　NCACHE 的管理维护　138
9.3.4　NCACHE 后端内容源服务器设置　139

第 10 章　Nginx 在国内知名网站中的应用案例　141
10.1　Nginx 反向代理与负载均衡类网站应用案例　142
10.1.1　Nginx 负载均衡在新浪播客中的应用　142
10.1.2　Nginx 负载均衡在金山逍遥网中的应用　146
10.2　Nginx+PHP 类网站应用案例　152
10.2.1　Nginx+PHP 在金山逍遥网 CMS 发布系统中的应用　152
10.2.2　Nginx+PHP 在某分类信息网站中的应用　154

第 11 章　Nginx 的非典型应用实例　171
11.1　用 HTTPS（SSL）构建一个安全的 Nginx Web 服务器　171
11.1.1　自行颁发不受浏览器信任的 SSL 证书　171
11.1.2　向 CA 机构申请颁发受浏览器信任的 SSL 证书　174
11.2　采用 Nginx 搭建 FLV 视频服务器　176
11.2.1　采用 Nginx 的 Flv Stream 模块搭建 HTTP 下载方式的 FLV 视频服务器　176
11.2.2　采用 Nginx 实现 FMS/Red5 流媒体视频服务器的负载均衡　177
11.3　Nginx+PHP+MySQL 在小内存 VPS 服务器上的优化　179
11.3.1　增加 swap 交换文件　180
11.3.2　Nginx 的主配置文件（nginx.conf）优化　180
11.3.3　PHP（FastCGI）的配置优化　182
11.3.4　MySQL 5.1 配置优化　182
11.4　采用 Nginx 搭建正向代理服务器　184

第 12 章　Nginx 的核心模块　185
12.1　主模块指令　185
12.1.1　daemon 指令　185
12.1.2　env 指令　186
12.1.3　debug_points 指令　186
12.1.4　error_log 指令　186
12.1.5　log_not_found 指令　187
12.1.6　include 指令　188
12.1.7　lock_file 指令　188
12.1.8　master_process 指令　188
12.1.9　pid 指令　189
12.1.10　ssl_engine 指令　189
12.1.11　timer_resolution 指令　189
12.1.12　try_files 指令　189
12.1.13　user 指令　191
12.1.14　worker_cpu_affinity 指令　191
12.1.15　worker_priority 指令　192
12.1.16　worker_processes 指令　192

12.1.17 worker_rlimit_core 指令 193
12.1.18 worker_rlimit_nofile 指令 193
12.1.18 worker_rlimit_sigpending 指令 193
12.1.19 working_directory 指令 193
12.2 主模块变量 193
12.3 事件模块指令 194
12.3.1 accept_mutex 指令 194
12.3.2 accept_mutex_delay 指令 194
12.3.3 debug_connection 指令 194
12.3.4 use 指令 195
12.3.5 worker_connections 指令 195

第 13 章 Nginx 的标准 HTTP 模块 197
13.1 HTTP 的核心模块 197
13.1.1 alias 指令 197
13.1.2 client_body_in_file_only 指令 198
13.1.3 client_body_in_single_buffer 指令 198
13.1.4 client_body_buffer_size 指令 198
13.1.5 client_body_temp_path 指令 198
13.1.6 client_body_timeout 指令 199
13.1.7 client_header_buffer_size 指令 199
13.1.8 client_header_timeout 指令 199
13.1.9 client_max_body_size 指令 199
13.1.10 default_type 指令 200
13.1.11 directio 指令 200
13.1.12 error_page 指令 201
13.1.13 if_modified_since 指令 201
13.1.14 index 指令 202
13.1.14 internal 指令 202
13.1.15 keepalive_timeout 指令 203
13.1.16 keepalive_requests 指令 203
13.1.17 large_client_header_buffers 指令 204
13.1.18 limit_except 指令 204
13.1.19 limit_rate 指令 204

13.1.20 limit_rate_after 指令 205
13.1.21 listen 指令 205
13.1.22 location 指令 206
13.1.24 log_not_found 指令 207
13.1.25 log_subrequest 指令 207
13.1.26 msie_padding 指令 208
13.1.27 msie_refresh 指令 208
13.1.28 open_file_cache 指令 208
13.1.29 open_file_cache_errors 指令 209
13.1.30 open_file_cache_min_uses 指令 209
13.1.31 open_file_cache_valid 指令 209
13.1.32 optimize_server_names 指令 209
13.1.33 port_in_redirect 指令 210
13.1.34 recursive_error_pages 指令 210
13.1.35 resolver 指令 210
13.1.36 resolver_timeout 指令 210
13.1.37 root 指令 211
13.1.38 satisfy_any 指令 211
13.1.39 send_timeout 指令 211
13.1.40 sendfile 指令 211
13.1.41 server 指令 212
13.1.42 server_name 指令 212
13.1.43 server_name_in_redirect 指令 214
13.1.44 server_names_hash_max_size 指令 214
13.1.45 server_names_hash_bucket_size 指令 214
13.1.46 server_tokens 指令 215
13.1.47 tcp_nodelay 指令 215
13.1.48 tcp_nopush 指令 215
13.1.49 try_files 指令 215
13.1.50 types 指令 216
13.1.51 HTTP 核心模块中可以使用的变量 216
13.2 HTTP Upstream 模块 219
13.3 Http Access 模块 219
13.3.1 allow 指令 219

13.3.2 deny 指令 219
13.4 HTTP Auth Basic 模块 220
13.4.1 auth_basic 指令 220
13.4.2 auth_basic_user_file 221
13.5 Http Autoindex 模块 221
13.5.1 autoindex 指令 221
13.5.2 autoindex_exact_size 指令 222
13.5.3 autoindex_localtime 指令 222
13.6 Http Browser 模块 222
13.6.1 ancient_browser 指令 223
13.6.2 ancient_browser_value 指令 223
13.6.3 modern_browser 指令 223
13.6.4 modern_browser_value 指令 225
13.7 Http Charset 模块 226
13.7.1 charset 指令 226
13.7.2 charset_map 指令 226
13.7.3 override_charset 指令 228
13.7.4 source_charset 指令 228
13.8 Http Empty Gif 模块 228
13.8.1 empty_gif 指令 228
13.9 Http Fcgi 模块 228
13.9.1 fastcgi_buffers 指令 229
13.9.2 fastcgi_buffer_size 指令 230
13.9.3 fastcgi_cache 指令 230
13.9.4 fastcgi_cache_key 指令 230
13.9.5 fastcgi_cache_methods 指令 230
13.9.6 fastcgi_index 指令 231
13.9.7 fastcgi_hide_header 指令 231
13.9.8 fastcgi_ignore_client_abort 指令 231
13.9.9 fastcgi_intercept_errors 指令 231
13.9.10 fastcgi_max_temp_file_size 指令 232
13.9.11 fastcgi_param 指令 232
13.9.12 fastcgi_pass 指令 233
13.9.13 fastcgi_pass_header 指令 234

13.9.14　fastcgi_read_timeout 指令　234
13.9.15　fastcgi_redirect_errors 指令　234
13.9.16　fastcgi_split_path_info 指令　234
13.10　Geo 模块　235
13.10.1　geo 指令　236
13.11　Gzip 模块　237
13.11.1　gzip 指令　238
13.11.2　gzip_buffers 指令　238
13.11.3　gzip_comp_level 指令　239
13.11.4　gzip_min_length 指令　239
13.11.5　gzip_http_version 指令　239
13.11.6　gzip_proxied 指令　240
13.11.7　gzip_types 指令　240
13.12　Http Headers 模块　241
13.12.1　add_header 指令　242
13.12.2　expires 指令　242
13.13　Http Index 模块　243
13.13.1　index 指令　243
13.14　Http Referer 模块　243
13.14.1　valid_referers 指令　244
13.15　Http Limit Zone 模块　244
13.15.1　limit_zone 指令　245
13.15.2　limit_conn 指令　245
13.16　Http Limit Req 模块　246
13.16.1　limit_req_zone 指令　246
13.16.2　limit_req 指令　247
13.17　Http Log 模块　248
13.17.1　access_log 指令　248
13.17.2　log_format 指令　249
13.17.3　log_format_combined 指令　249
13.17.4　open_log_file_cache 指令　250
13.18　Http Map 模块　250
13.18.1　map 指令　251
13.18.2　map_hash_max_size 指令　252

13.18.3　map_hash_bucket_size 指令　252
13.19　Http Memcached 模块　252
13.19.1　memcached_pass 指令　254
13.19.2　memcached_connect_timeout 指令　255
13.19.3　memcached_read_timeout 指令　255
13.19.4　memcached_send_timeout 指令　255
13.19.5　memcached_buffer_size 指令　256
13.19.6　memcached_next_upstream 指令　256
13.19.7　Http Memcached 模块中的变量　256
13.19.8　第三方的 Memcached 模块　256
13.20　Http Proxy 模块　257
13.20.1　proxy_buffer_size 指令　257
13.20.2　proxy_buffering 指令　257
13.20.3　proxy_buffers 指令　258
13.20.4　proxy_busy_buffers_size 指令　258
13.20.5　proxy_cache 相关指令集　259
13.20.6　proxy_connect_timeout 指令　259
13.20.7　proxy_headers_hash_bucket_size 指令　259
13.20.8　proxy_headers_hash_max_size 指令　259
13.20.9　proxy_hide_header 指令　260
13.20.10　proxy_ignore_client_abort 指令　260
13.20.11　proxy_ignore_headers 指令　260
13.20.12　proxy_intercept_errors 指令　261
13.20.13　proxy_max_temp_file_size 指令　261
13.20.14　proxy_method 指令　261
13.20.15　proxy_next_upstream 指令　262
13.20.16　proxy_pass 指令　263
13.20.17　proxy_pass_header 指令　264
13.20.18　proxy_pass_request_body 指令　265
13.20.19　proxy_pass_request_headers 指令　265
13.20.20　proxy_redirect 指令　265
13.20.21　proxy_read_timeout 指令　266
13.20.22　proxy_redirect_errors 指令　267
13.20.23　proxy_send_lowat 指令　267

13.20.24　proxy_send_timeout 指令　267
13.20.25　proxy_set_body 指令　268
13.20.26　proxy_set_header 指令　268
13.20.27　proxy_store 指令　269
13.20.28　proxy_store_access 指令　270
13.20.29　proxy_temp_file_write_size 指令　271
13.20.30　proxy_temp_path 指令　271
13.20.31　proxy_upstream_fail_timeout 指令　271
13.20.32　proxy_upstream_max_fails 指令　271
13.20.33　Http Proxy 模块的变量　272
13.21　Http Rewrite 模块　272
13.22　Http SSI 模块　272
13.22.1　ssi 指令　273
13.22.2　ssi_silent_errors 指令　273
13.22.3　ssi_types 指令　273
13.22.4　ssi_value_length 指令　274
13.22.5　SSI 命令　274
13.22.6　SSI 变量　276
13.27　Http Userid 模块　277
13.27.1　userid 指令　277
13.27.2　userid_domain 指令　277
13.27.3　userid_expires 指令　278
13.27.4　userid_name 指令　278
13.27.5　userid_p3p 指令　278
13.27.6　userid_path 指令　279
13.27.6　userid_service 指令　279

第 14 章　Nginx 的其他 HTTP 模块　280
14.1　Http Addition 模块　280
14.1.1　add_before_body　281
14.1.2　add_after_body　281
14.1.3　addition_types　281
14.2　Embedded Perl 模块　281
14.2.1　perl 指令　283

14.2.2　perl_modules 指令　283
14.2.3　perl_require 指令　283
14.2.4　perl_set 指令　283
14.2.5　从 SSI 调用 Perl 脚本　284
14.3　Flv Stream 模块　286
14.3　flv 指令　287
14.4　Http Gzip Static 模块　287
14.4.1　gzip_static 指令　287
14.4.2　gzip_http_version 指令　287
14.4.3　gzip_proxied 指令　288
14.5　Http Random Index 模块　288
14.5.1　random_index 指令　288
14.6　Http Geo IP 模块　288
14.6.1　geoip_country 指令　289
14.6.2　geoip_city 模块　289
14.7　Http RealIp 模块　290
14.7.1　set_real_ip_from 指令　290
14.7.2　real_ip_header 指令　291
14.8　Http SSL 模块　291
14.8.1　在多个 server{……}虚拟主机中使用通配符 SSL 证书　294
14.8.2　ssl 指令　294
14.8.3　ssl_certificate 指令　294
14.8.4　ssl_certificate_key 指令　295
14.8.5　ssl_client_certificate 指令　295
14.8.6　ssl_dhparam 指令　295
14.8.7　ssl_ciphers 指令　295
14.8.8　ssl_crl 指令　296
14.8.9　ssl_prefer_server_ciphers 指令　296
14.8.10　ssl_protocols 指令　296
14.8.11　ssl_verify_client 指令　296
14.8.12　ssl_verify_depth 指令　296
14.8.13　ssl_session_cache 指令　297
14.8.14　ssl_session_timeout 指令　297

14.8.15 ssl_engine 指令 298
14.9 Http Stub Status 模块 298
14.9.1 stub_status 指令 299
14.10 Http Sub 模块 299
14.10.1 sub_filter 指令 300
14.10.2 sub_filter_once 指令 300
14.10.3 sub_filter_types 指令 300
14.11 Http Dav 模块 300
14.11.1 dav_access 指令 301
14.11.2 dav_methods 指令 301
14.11.3 create_full_put_path 指令 301
14.12 Google Perftools 模块 302
14.12.1 google_perftools_profiles 指令 302
14.13 Http XSLT 模块 302
14.13.1 xslt_entities 指令 303
14.13.2 xslt_stylesheet 指令 303
14.13.3 xslt_types 指令 304
14.14 Http Secure Link 模块 304
14.14.1 secure_link_secret 指令 304
14.14.2 $secure_link 变量 305
14.15 Http Image Filter 模块 306
14.15.1 image_filter 306
14.15.2 image_filter_buffer 306
14.15.3 image_filter_jpeg_quality 307
14.15.4 image_filter_transparency 307

第 15 章 Nginx 的邮件模块 301

15.1 Nginx 邮件核心模块 303
15.2 Nginx 邮件认证模块 307
15.3 Nginx 邮件代理模块 308
15.4 Nginx 邮件 SSL 模块 310
15.5 Nginx 邮件模块配置实例 312

第1章 Nginx简介

1.1 常用的Web服务器简介

Web服务器也称为WWW（WORLD WIDE WEB）服务器、HTTP服务器，其主要功能是提供网上信息浏览服务。

Unix 和Linux平台下的常用Web服务器有Apache、Nginx、Lighttpd、Tomcat、IBM WebSphere等，其中应用最广泛的是Apache。而Windows NT/2000/2003平台下最常用的服务器则是微软公司的IIS（Internet Information Server）。

1.1.1 Apache服务器

Apache仍然是世界上用得最多的Web服务器，市场占有率达60%左右。它源于NCSAhttpd服务器，在NCSA WWW服务器项目停止后，那些使用NCSA WWW服务器的人们开始交换用于此服务器的补丁，这也是Apache名称的由来(pache补丁)。世界上很多著名的网站都是Apache的用户，它的优势主要在于源代码开放、有一支开放的开发队伍、支持跨平台的应用（可以运行在几乎所有的Unix、Windows、Linux系统平台上），以及其可移植性等。Apache的模块支持非常丰富，虽在速度、性能上不及其他轻量级 Web 服务器，但是属于重量级产品，所消耗的内存也比其他Web服务器要高。

官方网站：*http://httpd.apache.org/*。

www.TopSage.com

1.1.2 Lighttpd 服务器

Lighttpd 是由一个德国人写的开源软件，其目标是提供一个专门针对高性能网站，安全、快速、兼容性好并且灵活的 Web Server 环境。它具有内存开销低、CPU 占用率低、效能好，以及模块丰富等特点。支持 FastCGI 、CGI、Auth、输出压缩（output compress）、URL 重写及 Alias 等重要功能。Lighttpd 跟 Nginx 一样，也是一款轻量级 Web 服务器，是 Nginx 的竞争对手之一。

官方网站：*http://www.lighttpd.net/*。

1.1.3 Tomcat 服务器

Tomcat 是一个开放源代码、运行 servlet 和 JSP Web 应用软件的基于 Java 的 Web 应用软件容器。Tomcat Server 是根据 servlet 和 JSP 规范执行的，因此也可以说 Tomcat Server 实行了 Apache-Jakarta 规范，且比绝大多数商业应用软件服务器要好。但是，Tomcat 对静态文件、高并发的处理比较弱。

官方网站：*http://tomcat.apache.org*。

1.1.4 IBM WebSphere 服务器

WebSphere Application Server 是一种功能完善、开放的 Web 应用程序服务器，是 IBM 电子商务计划的核心部分，它基于 Java 的应用环境，建立、部署和管理 Internet 和 Intranet Web 应用程序。这一整套产品目前已进行了扩展，以适应 Web 应用程序服务器的需要，范围从简单到高级，直到企业级。据 IBM 官方网站介绍，有 10 000 多个企业正在使用 IBM WebSphere，相对于其他流行的 Web 服务器而言，应用的数量很少。

官方网站：*http://www.ibm.com/developerworks/cn/websphere/*。

1.1.5 Microsoft IIS

Microsoft 的 Web 服务器产品为 Internet Information Server（IIS），IIS 是允许在公共 Intranet 或 Internet 上发布信息的 Web 服务器。它是目前最流行的 Web 服务器产品，很多著名的网站都是建立在 IIS 平台上的。IIS 提供了一个图形界面的管理工具，称为 Internet 服务管理器，可用于监视配置和控制 Internet 服务。

IIS 是一种 Web 服务组件，其中包括 Web 服务器、FTP 服务器、NNTP 服务器和 SMTP 服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送等方面，它使得在网络（包括互联网和局

www.TopSage.com

大家网 TopSage.com

域网）上发布信息成了一件很容易的事。它提供 ISAPI（Intranet Server API）作为扩展 Web 服务器功能的编程接口；同时，它还提供一个 Internet 数据库连接器，可以实现对数据库的查询和更新。

IIS 只能运行在 Microsoft Windows 平台、Linux/Unix 平台上，因此须要购买商业的 Windows Server 操作系统。

1.2 Nginx 的发展

Nginx（“engine x”）是俄罗斯人 Igor Sysoev（伊戈尔·塞索耶夫）编写的一款高性能的 HTTP 和反向代理服务器。Nginx 能够选择高效的 epoll（Linux 2.6 内核）、kqueue（FreeBSD）、eventport（Solaris 10）作为网络 I/O 模型，在高连接并发的情况下，Nginx 是 Apache 服务器不错的替代品，它能够支持高达 50 000 个并发连接数的响应，而内存、CPU 等系统资源消耗却非常低，运行非常稳定。

Nginx 已经在俄罗斯最大的门户网站—— Rambler Media（www.rambler.ru）上运行了 3 年时间，同时俄罗斯超过 20%的虚拟主机平台采用 Nginx 作为反向代理服务器。

在国内，已经有新浪博客、新浪播客、网易新闻、六间房、56.com、Discuz!官方论坛、水木社区、豆瓣、YUPOO 相册、海内 SNS、迅雷在线等多家网站使用 Nginx 作为 Web 服务器或反向代理服务器。

图 1-1 是 Netcraft 公司统计的从 1995 年 8 月至 2009 年 1 月各 Web 服务器的市场占有率曲线图。

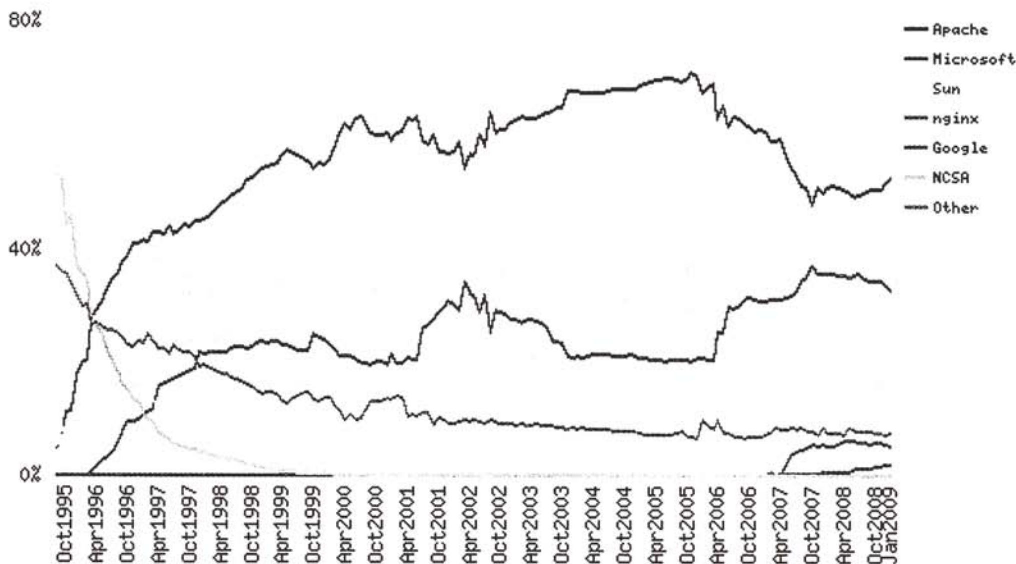


图 1-1 Web 服务器的市场占有率曲线图

www.TopSage.com

2009 年 1 月，对 185 497 213 个网站进行了抽样调查，发现除去 Google 自己开发的仅供自己使用的 GWS、GFE 服务器外，排在前两位的分别是 Apache、Microsoft IIS，而 Nginx 已经超过 Lighttpd，位居第三，详见表 1-1。

表 1-1 Nginx 的市场占有率排名

Web 服务器	2008 年 12 月	占有率	2009 年 1 月	占有率	占有率变化
Apache	95 678 052	51.24%	96 947 298	52.26%	1.02%
Microsoft IIS	63 126 940	33.81%	61 038 371	32.91%	–0.90%
Google	10 455 103	5.60%	9 868 819	5.32%	–0.28%
Nginx	3 354 329	1.80%	3 462 551	1.87%	0.07%
Lighttpd	3 046 333	1.63%	2 989 416	1.61%	–0.02%

1.3 选择 Nginx 的理由

选择 Nginx 有如下一些理由。

1.3.1 它可以高并发连接

官方测试 Nginx 能够支撑 5 万并发连接，在实际生产环境中可支撑 2～4 万并发连接数。这得益于 Nginx 使用了最新的 epoll（Linux 2.6 内核）和 kqueue（freebsd）网络 I/O 模型，而 Apache 使用的则是传统的 select 模型，其比较稳定的 Prefork 模式为多进程模式，需要经常派生子进程，所消耗的 CPU 等服务器资源要比 Nginx 高得多。

笔者曾完成 6 台 Web Server 从 Apache 到 Nginx 服务器的迁移（这 6 台 Web Server 搭建的是一个日均 2 500 万 PV 的分类信息网站，迁移前每台服务器的平均系统负载为 50～60、CPU 使用率为 70%～90%，迁移后平均系统负载为 1～4，CPU 使用率为 20%～40%，见图 1-2。

```
top - 17:59:33 up 10 days,  8:10,  3 users,  load average: 2.69, 2.17, 2.57
Tasks: 300 total,   1 running, 299 sleeping,   0 stopped,   0 zombie
Cpu(s): 30.1% us,  5.0% sy,  0.0% ni, 64.1% id,  0.1% wa,  0.1% hi,  0.7% si
Mem:   8230016k total,  3850368k used,  2379648k free,   588616k buffers
Swap:  2031608k total,      508k used,  2031100k free,  1755208k cached
```

图 1-2 迁移到 Nginx 后的系统负载与 CPU 使用率图

由此可见，处理大量连接的读写，Apache 所采用的 select 网络 I/O 模型非常低效。

Libevent（一个事件触发的网络库，适用于 Windows、Linux、bsd 等多种平台，内部使用 select、epoll、kqueue 等系统调用管理事件机制）的一张测试结果图，对 select、epoll、kqueue 等作了清

www.TopSage.com

晰的对比，结果自然是 epoll（Linux 2.6 内核）和 kqueue（freebsd）胜出，见图 1-3。

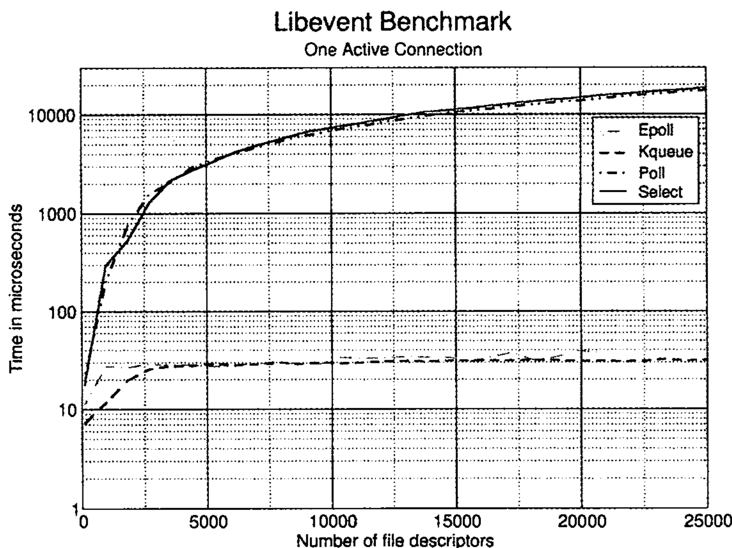


图 1-3 epoll、kqueue、select 等网络 I/O 模型性能测试对比图

图 1-4 显示了在实际的生产环境中，Nginx 支撑高达 28 000 的活动并发连接数。

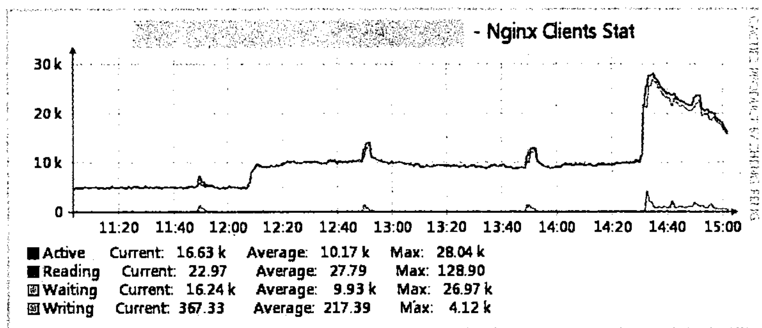


图 1-4 生产环境下的 Nginx 活动并发连接数据统计图

1.3.2 内存消耗少

Nginx + PHP（FastCGI）服务器在 3 万并发连接下，开启的 10 个 Nginx 进程消耗 150MB 内存（15MB×10=150MB），开启的 64 个 php-cgi 进程消耗 1280MB 内存（20MB×64=1280MB），加上系统自身消耗的内存，总共消耗不到 2GB 的内存。如果服务器内存较小，完全可以只开启 25 个 php-cgi 进程，这样 php-cgi 消耗的总内存数才 500MB。用 Webbench 做压力测试，在 3 万

www.TopSage.com

大家网 TopSage.com

并发连接下，访问 Nginx + PHP（FastCGI）服务器的 PHP 程序，运行速度仍然飞快。

在实际的生产环境下，两台 Nginx + PHP5（FastCGI）服务器运行多个复杂性一般的纯 PHP 动态程序，从 Nginx 的日志可以统计出，单台 Nginx + PHP5（FastCGI）服务器处理 PHP 动态程序的能力已经超过“700 次请求/秒”（见图 1-5），相当于每天可以承受 6 000 万（700×60×60×24=60 480 000）的访问量，而服务器的系统负载也不算高（见图 1-6）。

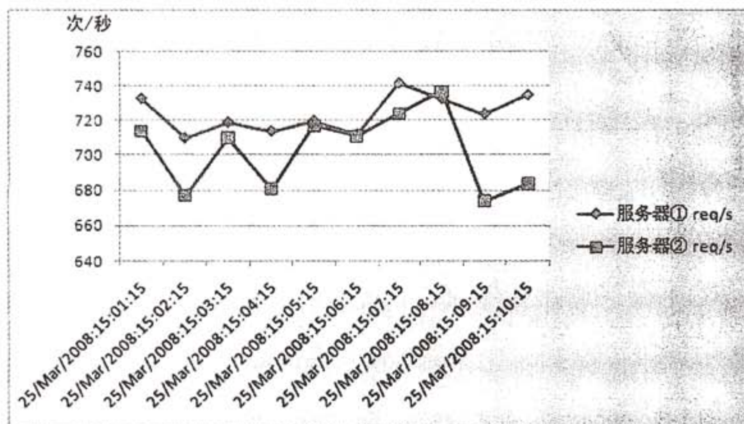
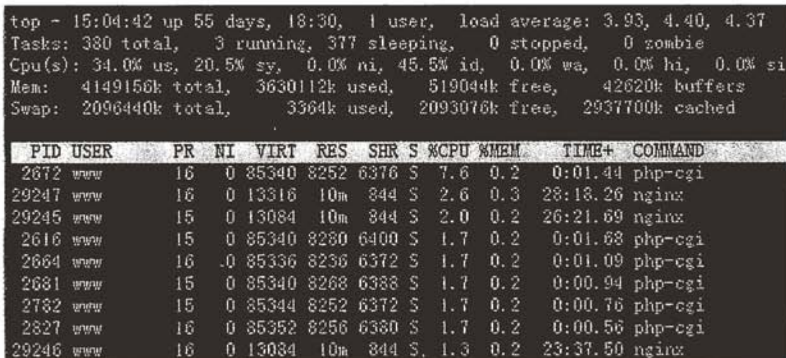


图 1-5 生产环境下的 Nginx+PHP 动态程序处理速度统计图



```
top - 15:04:42 up 55 days, 18:30,  1 user,  load average: 3.93, 4.40, 4.37
Tasks: 380 total,   3 running, 377 sleeping,   0 stopped,   0 zombie
Cpu(s): 34.0% us, 20.5% sy,  0.0% ni, 45.5% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   4149156k total,  3630112k used,   519044k free,    42620k buffers
Swap:  2096440k total,     3364k used,  2093076k free,  2937700k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2672 www       16   0 85340 8252 6376 S  7.6  0.2   0:01.44 php-cgi
29247 www       16   0 13316  10m  844 S  2.6  0.3  28:18.26 nginx
29245 www       15   0 13084  10m  844 S  2.0  0.2  26:21.69 nginx
 2616 www       15   0 85340 8280 6400 S  1.7  0.2   0:01.68 php-cgi
 2664 www       16   0 85336 8236 6372 S  1.7  0.2   0:01.09 php-cgi
 2681 www       15   0 85340 8268 6388 S  1.7  0.2   0:00.94 php-cgi
 2782 www       15   0 85344 8252 6372 S  1.7  0.2   0:00.76 php-cgi
 2827 www       16   0 85352 8256 6380 S  1.7  0.2   0:00.56 php-cgi
29246 www       16   0 13084  10m  844 S  1.3  0.2  23:37.50 nginx
```

图 1-6 生产环境下的 Nginx+PHP 系统负载与 CPU 使用率图

以下是这两台服务器的配置清单及运行的程序说明：

服务器①：DELL PowerEdge 1950（两个 Intel(R) Xeon(R) 双核 CPU 5120 @ 1.86GHz，4GB 内存）。

服务器②：DELL PowerEdge 1950（一个 Intel(R) Xeon(R) 双核 CPU 5140 @ 2.33GHz，4GB 内存）。

www.TopSage.com

Web 服务器：CentOS Linux 4.4 + Nginx 0.5.35 + PHP 5.2.6RC2（300 FastCGI Procees, unix-domain socket, with XCache）。

PHP 程序内容：大量 Memcached 读写操作，少量 MySQL 读操作，大量文件队列写操作。

请求数统计方式：从 Nginx 访问日志中，统计每分钟的第 15 秒共有多少条日志记录。

同等硬件环境下，Nginx 的处理能力相当于 Apache 的 5～10 倍。

1.3.3 成本低廉

购买 F5 BIG-IP、NetScaler 等硬件负载均衡交换机需要十多万甚至几十万人民币。而 Nginx 为开源软件，采用的是 2-clause BSD-like 协议，可以免费使用，并且可用于商业用途。

BSD 开源协议是一个给使用者很大自由的协议。协议指出可以自由使用、修改源代码，也可以将修改后的代码作为开源或专有软件再发布。当你发布使用了 BSD 协议的代码，或者以 BSD 协议代码为基础做二次开发时，须满足三个条件：

（1）如果再发布的产品中包含源代码，则源代码中必须带有原来代码中的 BSD 协议。

（2）如果再发布的是二进制类库/软件，则需要在类库/软件的文档和版权声明中包含原来代码中的 BSD 协议。

（3）不可以用开源代码的作者/机构名字和原来产品的名字做市场推广。

BSD 代码鼓励代码共享，但须尊重代码作者的著作权。BSD 由于允许使用者修改和重新发布代码，也允许使用或在 BSD 代码上开发商业软件，并进行发布和销售，因此它是对商业集成很友好的协议。很多的公司、企业在选用开源产品的时候都会首选 BSD 协议，因为可以完全控制这些第三方的代码，在必要的时候可以修改或二次开发。

Nginx 所采用的 2-clause BSD-like license 衍生自 BSD 协议，也就是删掉了 BSD 协议的第 3 个条件——“不可以用开源代码的作者/机构名字和原来产品的名字做市场推广”。

1.3.4 其他理由

配置文件非常简单

网络跟程序一样通俗易懂，即使非专业系统管理员也能看懂。

支持 Rewrite 重写规则

能够根据域名、URL 的不同，将 HTTP 请求分到不同的后端服务器群组。

www.TopSage.com

内置的健康检查功能

如果 Nginx Proxy 后端的某台 Web 服务器宕机了，不会影响前端访问。

节省带宽

支持 GZIP 压缩，可以添加浏览器本地缓存的 Header 头。

稳定性高

用于反向代理，宕机的概率微乎其微。

支持热部署

Nginx 支持热部署。它的启动特别容易，并且几乎可以 7 天×24 小时不间断地运行，即使运行数个月也不需要重新启动。你还能够在不间断服务的情况下，对软件版本进行升级。

1.4 Nginx 与 Apache、Lighttpd 的综合对比

表 1-2 是 Nginx 与 Apache、Lighttpd 的综合对比情况。

表 1-2 Nginx 与 Apache、Lighttpd 的综合对比

Web 服务器	Nginx	Apache	Lighttpd
反向代理	非常好	好	一般
Rewrite 规则	非常好	好	一般
FastCGI	好	差	非常好
热部署	支持	不支持	不支持
系统压力比较	很小	小	很大
稳定性	非常好	好	一般
安全性	一般	好	一般
技术资料	很少	非常多	一般
静态文件处理	非常好	一般	好
虚拟主机	支持	支持	支持
内存消耗	非常小	很大	非常小

通过表 1-2 可以看出，Nginx 在反向代理、Rewrite 规则、稳定性、静态文件处理、内存消耗等方面，表现出了很强的优势，选用 Nginx 取代传统的 Apache 服务器，将会获得多方面的性能提升。

www.TopSage.com

大家网
TopSage.com

第 2 章 Nginx 服务器的安装与配置

2.1 安装 Nginx 服务器所需要的系统资源

Nginx 是开源软件，您可以从其官方网站（*http://www.nginx.net/*）下载最新版本。Nginx 目前有 3 个版本：旧的稳定版（0.6.x）、最新的稳定版（0.7.x）和开发版（0.8.x）。0.8.x 开发版分支刚发布不久，Bug 会比较多，因此不建议用于生产环境。开发版一旦更新稳定下来，就会被加入稳定版分支。然而，新功能不一定会被加到旧的稳定版中去，所以，目前最合适使用的版本是 0.7.x。

从 0.7.52 版本开始，Nginx 官方网站开始提供 Windows 版本下载，Windows 版本的 Nginx 使用比较简单，只须下载完成后，将其解压缩到一个不包含空格的路径中，执行 nginx.exe 即可。但是，Windows 版本的 Nginx 性能要比 Linux/Unix 版本的 Nginx 差很多。本书重点介绍 Linux 环境下的 Nginx 编译安装。

一些 Linux 发行版和 BSD 的各个变种版本的安装包仓库中包含了编译后的二进制 Nginx 软件，很多预先编译好的安装包都比较陈旧，所以大多数情况下还是推荐直接从源码编译安装。

安装 Nginx 服务器之前，首先要安装一个 Linux/Unix 操作系统发行版，例如 Redhat、CentOS、Debian、Fedora Core、Gentoo、SUSE、Ubuntu、FreeBSD 等。

本书将以新浪、搜狐、网易、金山游戏官网等国内互联网公司最常用的 Linux 发行版——CentOS 为例，介绍 Nginx 的安装与使用。CentOS 是基于 RedHat Enterprise Linux 源代码重新编译、去除 RedHat 商标的产物，各种操作、使用和 RedHat 没有区别。CentOS 完全免费，修正了 RedHat 的很多 BUG，但 CentOS 不向用户提供技术支持，也不负任何商业责任。

www.TopSage.com

编译 Nginx 的要求如下：

磁盘空间：需要保证有 10MB 以上的剩余磁盘空间。Nginx 安装完毕后会占据 4MB 左右的磁盘空间，实际的磁盘空间需求会因编译设置和是否安装第三方模块而有所不同。

GCC 编译器及相关工具：GCC 全称为 GNU Compiler Collection， 是 GNU 社区推出的功能强大、性能优越的用于编程开发的自由编译器，是 GNU 的代表作品之一，目前可以编译的语言包括：C、C++、Objective-C、Fortran、Java 等。您必须确保您的操作系统安装有 GCC 编译器。另外，您还必须安装 Autoconf 和 Automake 工具，它们用于自动创建功能完善的 Makefile，当前大多数软件包都是用这一工具生成 Makefile 的，Nginx 也不例外。在 CentOS 系统下，您可以使用 yum 命令安装 GCC 编译器及相关工具：

```
yum -y install gcc gcc-c++ autoconf automake
```

模块依赖性：Nginx 的一些模块需要其他第三方库的支持，例如 gzip 模块需要 zlib 库，rewrite 模块需要 pcre 库，ssl 功能需要 openssl 库等。同样，如果是在 CentOS 系统下，我们可以使用 yum 命令安装或下载源码包编译安装这些模块依赖的库：

```
yum -y install zlib zlib-devel openssl openssl-devel pcre pcre-devel
```

2.2 Nginx 的下载

您可以访问 *http://www.nginx.net/*网站，该网站中的以下三行内容分别提供了 Nginx 0.6.x、0.7.x、0.8.x 三个版本分支的源码包或 Windows 二进制文件下载。

The development stable versions are nginx-0.8.x, nginx/Windows-0.8.x, the change log

The latest stable versions are nginx-0.7.x, nginx/Windows-0.7.x, the change log

The latest legacy stable version is nginx-0.6.x, the change log

如果您希望在 Windows 上运行 Nginx，你可以点击链接“nginx/Windows-0.8.x”或“nginx/Windows-0.7.x”下载 Windows 下的二进制版 ZIP 压缩包。如果您希望在 Linux/Unix 环境下运行 Nginx，您可以点击链接“nginx-0.8.x”、“nginx-0.7.x”、“nginx-0.6.x”下载以 tar.gz 格式压缩源码包。

2.3 Nginx 的安装

Nginx 从 0.7.52 版本开始有了官方的 Windows 版本，所以这里分别介绍 Nginx 在 Windows 下和 Linux 下的安装步骤。

www.TopSage.com

2.3.1 Nginx 在 Windows 环境下的安装

Nginx 在 Windows 下的安装比较简单，将下载下来的 nginx-0.x.xx.zip 文件解压缩到一个不包含空格的路径中，例如 d:\nginx，然后在"开始"→"运行"→"cmd"中执行以下 DOS 命令即可启动 Nginx：

```
d:
cd d:\nginx
start nginx
```

如果要对启动的 Nginx 进程进行控制，可以使用 DOS 命令：

```
nginx -s [ stop | quit | reopen | reload ]
```

2.3.2 Nginx 在 Linux 环境下的安装

Nginx 在 Linux 环境下可以通过编译源码的方式来安装，最简单的安装命令如下：

```
tar zxvf nginx-0.x.xx.tar.gz
cd nginx-0.x.xx
./configure
make
sudo make install
```

按照以上命令，Nginx 将被默认安装到/usr/local/nginx 目录下。您可以通过./configure --help 命令查看 Nginx 可选择的编译选项。

Nginx 的 configure 脚本支持以下选项：

--prefix=<path>——Nginx 安装路径。如果没有指定，默认为 /usr/local/nginx。

--sbin-path=<path>——Nginx 可执行文件安装路径。只能安装时指定，如果没有指定，默认为<prefix>/sbin/nginx。

--conf-path=<path>——在没有给定-c 选项下默认的 nginx.conf 的路径。如果没有指定，默认为<prefix>/conf/nginx.conf。

--pid-path=<path>——在 nginx.conf 中没有指定 pid 指令的情况下，默认的 Nginx.pid 的路径。如果没有指定，默认为 <prefix>/logs/nginx.pid。

--lock-path=<path>——nginx.lock 文件的路径。

--error-log-path=<path>——在 nginx.conf 中没有指定 error_log 指令的情况下，默认的错误日志的路径。如果没有指定，默认为 <prefix>/logs/error.log。

--http-log-path=<path>——在 nginx.conf 中没有指定 access_log 指令的情况下，默认的访问日

www.TopSage.com

大家网 TopSage.com

志的路径。如果没有指定，默认为 <prefix>/logs/access.log。

--user=<user>——在 nginx.conf 中没有指定 user 指令的情况下，默认的 Nginx 使用的用户。如果没有指定，默认为 nobody。

--group=<group>——在 nginx.conf 中没有指定 user 指令的情况下，默认的 Nginx 使用的组。如果没有指定，默认为 nobody。

--builddir=DIR——指定编译的目录。

--with-rtsig_module——启用 rtsig 模块。

--with-select_module（--without-select_module）——允许或不允许开启 SELECT 模式，如果 configure 没有找到更合适的模式，比如：kqueue（sun os）、epoll（linux kenel 2.6+）、rtsig（实时信号）或/dev/poll（一种类似 select 的模式，底层实现与 SELECT 基本相同，都是采用轮训方法），SELECT 模式将是默认安装模式。

--with-poll_module（--without-poll_module）——允许或不允许开启 POLL 模式，如果没有更合适的模式，比如：kqueue（sun os）、epoll（linux kenel 2.6+），则可以开启。

--with-http_ssl_module——开启 HTTP SSL 模块，使 Nginx 可以支持 HTTPS 请求。这个模块需要已经安装 OPENSSL，在 DEBIAN 上是 libssl。

--with-http_realip_module——启用 ngx_http_realip_module。

--with-http_addition_module——启用 ngx_http_addition_module。

--with-http_sub_module——启用 ngx_http_sub_module。

--with-http_dav_module——启用 ngx_http_dav_module。

--with-http_flv_module——启用 ngx_http_flv_module。

--with-http_stub_status_module——启用“server status”页。

--without-http_charset_module——禁用 ngx_http_charset_module。

--without-http_gzip_module——禁用 ngx_http_gzip_module。如果启用，需要 zlib。

--without-http_ssi_module——禁用 ngx_http_ssi_module。

--without-http_userid_module——禁用 ngx_http_userid_module。

--without-http_access_module——禁用 ngx_http_access_module。

--without-http_auth_basic_module——禁用 ngx_http_auth_basic_module。

--without-http_autoindex_module——禁用 ngx_http_autoindex_module。

--without-http_geo_module——禁用 ngx_http_geo_module。

www.TopSage.com

--without-http_map_module——禁用 ngx_http_map_module。

--without-http_referer_module——禁用 ngx_http_referer_module。

--without-http_rewrite_module——禁用 ngx_http_rewrite_module。如果启用，需要 PCRE。

--without-http_proxy_module——禁用 ngx_http_proxy_module。

--without-http_fastcgi_module——禁用 ngx_http_fastcgi_module。

--without-http_memcached_module——禁用 ngx_http_memcached_module。

--without-http_limit_zone_module——禁用 ngx_http_limit_zone_module。

--without-http_empty_gif_module——禁用 ngx_http_empty_gif_module。

--without-http_browser_module——禁用 ngx_http_browser_module。

--without-http_upstream_ip_hash_module——禁用 ngx_http_upstream_ip_hash_module。

--with-http_perl_module——启用 ngx_http_perl_module。

--with-perl_modules_path=PATH——指定 perl 模块的路径。

--with-perl=PATH——指定 perl 执行文件的路径。

--http-log-path=PATH——指定 http 默认访问日志的路径。

--http-client-body-temp-path=PATH——指定 http 客户端请求缓存文件存放目录的路径。

--http-proxy-temp-path=PATH——指定 http 反向代理缓存文件存放目录的路径。

--http-fastcgi-temp-path=PATH——指定 http FastCGI 缓存文件存放目录的路径。

--without-http——禁用 HTTP server。

--with-mail——启用 IMAP4/POP3/SMTP 代理模块。

--with-mail_ssl_module——启用 ngx_mail_ssl_module。

--with-cc=PATH——指定 C 编译器的路径。

--with-cpp=PATH——指定 C 预处理器的路径。

--with-cpu-opt=CPU——为特定的 CPU 编译，有效的值包括：pentium、pentiumpro、pentium3、pentium4、athlon、opteron、amd64、sparc32、sparc64、ppc64。

--without-pcre——禁止 PCRE 库的使用。同时也会禁止 HTTP rewrite 模块。在“location”配置指令中的正则表达式也需要 PCRE。

--with-pcre=DIR——指定 PCRE 库的源代码的路径。

--with-pcre-opt=OPTIONS——设置 PCRE 的额外编译选项。

www.TopSage.com

--with-md5=DIR——设置 MD5 库的源代码路径。

--with-md5-opt=OPTIONS——设置 MD5 库的额外编译选项。

--with-md5-asm——使用 MD5 汇编源码。

--with-sha1=DIR——设置 sha1 库的源代码路径。

--with-sha1-opt=OPTIONS——设置 sha1 库的额外编译选项。

--with-sha1-asm——使用 sha1 汇编源码。

--with-zlib=DIR——设置 zlib 库的源代码路径。

--with-zlib-opt=OPTIONS——设置 zlib 库的额外编译选项。

--with-zlib-asm=CPU——zlib 针对 CPU 的优化，合法的值是：pentium、pentiumpro。

--with-openssl=DIR——设置 OpenSSL 库的源代码路径。

--with-openssl-opt=OPTIONS——设置 OpenSSL 库的额外编译选项。

--with-debug——启用调试日志。

--add-module=PATH——添加一个在指定路径中能够找到的第三方模块。

在不同版本间，选项可能会有些许变化，请总是使用 ./configure --help 命令来检查当前的选项列表。

一个自定义编译选项的示例如代码 2-1 所示：

代码 2-1

```
./configure
  --prefix=/usr \
  --sbin-path=/usr/sbin/nginx \
  --conf-path=/etc/nginx/nginx.conf \
  --error-log-path=/var/log/nginx/error.log \
  --pid-path=/var/run/nginx/nginx.pid  \
  --lock-path=/var/lock/nginx.lock \
  --user=nginx \
  --group=nginx \
  --with-http_ssl_module \
  --with-http_flv_module \
  --with-http_gzip_static_module \
  --http-log-path=/var/log/nginx/access.log \
  --http-client-body-temp-path=/var/tmp/nginx/client/ \
  --http-proxy-temp-path=/var/tmp/nginx/proxy/ \
  --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/
```


www.TopSage.com

大家网 TopSage.com

2.4 Nginx 的启动、停止、平滑重启

在 Linux 下，Nginx 服务主要的操作就是启动、停止和平滑重启。

2.4.1 Nginx 的启动

启动 Nginx，可以执行以下命令。假设 Nginx 安装在/usr/local/nginx/目录中，那么启动 Nginx 的命令就是：

```
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

参数“-c”指定了配置文件的路径，如果不加“-c”参数，Nginx 会默认加载其安装目录的 conf 子目录中的 nginx.conf 文件，在本例中即：/usr/local/nginx/sbin/nginx/conf/nginx.conf。

2.4.2 Nginx 的停止

Nginx 的停止方法有很多种，一般通过发送系统信号给 Nginx 主进程的方式来停止 Nginx。

我们可以通过 ps 命令来查找 Nginx 的主进程号：

```
ps -ef | grep nginx
```

这时候屏幕会显示如图 2-1 所示的信息。

```
root      28881     1  0 01:32 ?        00:00:00 nginx: master process /usr/local/webserver/nginx/sbin/nginx
www       28882 28881  0 01:32 ?        00:00:00 nginx: worker process
www       28883 28881  0 01:32 ?        00:00:00 nginx: worker process
www       28884 28881  0 01:32 ?        00:00:00 nginx: worker process
www       28885 28881  0 01:32 ?        00:00:00 nginx: worker process
root      28892 28848  0 01:34 pts/0    00:00:00 grep nginx
```

图 2-1 查看 Nginx 进程 ID

从图 2-1 中可以看到，1 个 Nginx 进程的备注信息为“master process”，表示它为主进程，另外的 4 个进程备注信息为“worker process”，表示它们为子进程。28881 为主进程号。

如果在 nginx.conf 配置文件中指定了 pid 文件存放的路径（例如：pid /usr/local/webserver/nginx/logs/nginx.pid;），该文件中存放的就是 Nginx 当前的主进程号。如果没有指定 pid 文件存放的路径，nginx.pid 文件默认存放在 Nginx 安装目录的 logs 目录下。所以，我们可以直接通过以下命令来完成平滑重启，省下寻找 Nginx 主进程号的步骤：

```
kill - 信号类型 `/usr/local/webserver/nginx/logs/nginx.pid`
```

（1）从容停止 Nginx。

```
kill - QUIT Nginx 主进程号
```

或

```
kill - QUIT `/usr/local/webserver/nginx/logs/nginx.pid`
```

（2）快速停止 Nginx。

```
kill - TERM Nginx 主进程号

kill - TERM `/usr/local/webserver/nginx/logs/nginx.pid`
```

或

```
kill - INT Nginx 主进程号

kill - INT `/usr/local/webserver/nginx/logs/nginx.pid`
```

（3）强制停止所有 Nginx 进程。

```
pkill -9 nginx
```

2.5 Nginx 的平滑重启

如果改变了 Nginx 的配置文件（nginx.conf），想重启 Nginx，同样可以通过发送系统信号给 Nginx 主进程的方式来进行。不过，重启之前，要确认 Nginx 配置文件（nginx.conf）的语法是正确的，否则 Nginx 将不会加载新的配置文件。通过以下命令可以判断 Nginx 配置文件是否正确：

```
/usr/local/webserver/nginx/sbin/nginx -t -c
/usr/local/webserver/nginx/conf/nginx.conf
```

如果配置文件不正确，屏幕将会提示配置文件的第几行出错：

[emerg]: unknown directive "abc" in /usr/local/webserver/nginx/conf/nginx.conf:55

configuration file /usr/local/webserver/nginx/conf/nginx.conf test failed

如果配置文件正确，屏幕将提示以下两行信息：

the configuration file /usr/local/webserver/nginx/conf/nginx.conf syntax is ok

configuration file /usr/local/webserver/nginx/conf/nginx.conf test is successful

这时候，就可以平滑重启 Nginx 了。

```
kill -HUP Nginx 主进程号

kill -HUP `/usr/local/webserver/nginx/logs/nginx.pid`
```

当 Nginx 接收到 HUP 信号时，它会尝试先解析配置文件（如果指定配置文件，就使用指定的，否则使用默认的），如果成功，就应用新的配置文件（例如，重新打开日志文件或监听的套

www.TopSage.com

接字）。之后，Nginx 运行新的工作进程并从容关闭旧的工作进程。通知工作进程关闭监听套接字，但是继续为当前连接的客户提供服务。所有客户端的服务完成后，旧的工作进程被关闭。如果新的配置文件应用失败，Nginx 将继续使用旧的配置进行工作。

2.6 Nginx 的信号控制

在上一节中，我们使用了信号来控制 Nginx 停止、平滑重启，Nginx 支持以下几种信号：

- TERM, INT 快速关闭；
- QUIT 从容关闭；
- HUP 平滑重启，重新加载配置文件；
- USR1 重新打开日志文件，在切割日志时用途较大；
- USR2 平滑升级可执行程序；
- WINCH 从容关闭工作进程。

2.7 Nginx 的平滑升级

当需要将正在运行中的 Nginx 升级、添加/删除服务器模块时，可以在不中断服务的情况下，使用新版本、重编译的 Nginx 可执行程序替换旧版本的可执行程序。步骤如下：

（1）使用新的可执行程序替换旧的可执行程序，对于编译安装的 Nginx，可以将新版本编译安装到旧版本的 Nginx 安装路径中。替换之前，您最好备份一下旧的可执行文件。

（2）发送以下指令：

```
kill -USR2 旧版本的Nginx主进程号
```

（3）旧版本 Nginx 的主进程将重命名它的.pid 文件为.oldbin（例如：/usr/local/webserver/nginx/logs/nginx.pid.oldbin），然后执行新版本的 Nginx 可执行程序，依次启动新的主进程和新的工作进程。

```
PID    PPID USER   %CPU  VSZ WCHAN  COMMAND
33126      1 root   0.0  1164 pause  nginx: master process /usr/local/nginx/sbin/nginx
33135 33126 nobody  0.0  1380 kqread nginx: worker process is shutting down (nginx)
36264 33126 root    0.0  1148 pause  nginx: master process /usr/local/nginx/sbin/nginx
36265 36264 nobody  0.0  1364 kqread nginx: worker process (nginx)
```


www.TopSage.com

大家网 TopSage.com

36266 36264 nobody 0.0 1364 kqread nginx: worker process (nginx)

36267 36264 nobody 0.0 1364 kqread nginx: worker process (nginx)

（4）此时，新、旧版本的 Nginx 实例会同时运行，共同处理输入的请求。要逐步停止旧版本的 Nginx 实例，你必须发送 WINCH 信号给旧的主进程，然后，它的工作进程就将开始从容关闭：

```
kill -WINCH 旧版本的 Nginx 主进程号
```

（5）一段时间后，旧的工作进程（worker process）处理了所有已连接的请求后退出，仅由新的工作进程来处理输入的请求了：

PID	PPID	USER	%CPU	VSZ	WCHAN	COMMAND
33126	1	root	0.0	1164	pause	nginx: master process /usr/local/nginx/sbin/nginx
36264	33126	root	0.0	1148	pause	nginx: master process /usr/local/nginx/sbin/nginx
36265	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)
36266	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)
36267	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)

（6）这时候，我们可以决定是使用新版本，还是恢复到旧版本：

kill -HUP 旧的主进程号：Nginx 将在不重载配置文件的情况下启动它的工作进程；

kill -QUIT 新的主进程号：从容关闭其工作进程（worker process）；

kill -TERM 新的主进程号：强制退出；

kill 新的主进程号或旧的主进程号：如果因为某些原因新的工作进程不能退出，则向其发送 kill 信号。

新的主进程退出后，旧的主进程会移除 .oldbin 前缀，恢复为它的 .pid 文件，这样，一切就都恢复到升级之前了。如果尝试升级成功，而你也希望保留新的服务器时，可发送 QUIT 信号给旧的主进程，使其退出而只留下新的服务器运行：

PID	PPID	USER	%CPU	VSZ	WCHAN	COMMAND
36264	1	root	0.0	1148	pause	nginx: master process /usr/local/nginx/sbin/nginx
36265	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)
36266	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)
36267	36264	nobody	0.0	1364	kqread	nginx: worker process (nginx)

www.TopSage.com

大家网
TopSage.com

第 3 章 Nginx 的基本配置与优化

3.1 Nginx 的完整配置示例

Nginx 的配置文件默认在 Nginx 程序安装目录的 conf 二级目录下，主配置文件为 nginx.conf，假设您的 Nginx 安装在/usr/local/webserver/nginx/目录下，那么默认的主配置文件则为/usr/local/webserver/nginx/nginx.conf，代码 3-1 是 Nginx 作为 Web Server 的完整配置示例。

代码 3-1

```
# 使用的用户和组
user  www www;
# 指定工作衍生进程数（一般等于 CPU 的总核数或总核数的两倍，例如两个四核 CPU，则总核数为 8）
worker_processes 8;
# 指定错误日志存放的路径，错误日志记录级别可选项为：[ debug | info | notice | warn | error
| crit ]
error_log  /data1/logs/nginx_error.log  crit;
# 指定 pid 存放的路径
pid        /usr/local/webserver/nginx/nginx.pid;

# 指定文件描述符数量
worker_rlimit_nofile 51200;

events
{
  # 使用的网络 I/O 模型，Linux 系统推荐采用 epoll 模型，FreeBSD 系统推荐采用 kqueue 模型
  use epoll;
```


www.TopSage.com

```
  # 允许的连接数
  worker_connections 51200;
}

http
{
  include       mime.types;
  default_type application/octet-stream;
  # 设置使用的字符集，如果一个网站有多种字符集，请不要随便设置，应让程序员在 HTML 代码中通过 Meta
    标签设置
  #charset  gb2312;

  server_names_hash_bucket_size 128;
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;

  # 设置客户端能够上传的文件大小
  client_max_body_size 8m;

  sendfile on;
  tcp_nopush     on;

  keepalive_timeout 60;

  tcp_nodelay on;

  fastcgi_connect_timeout 300;
  fastcgi_send_timeout 300;
  fastcgi_read_timeout 300;
  fastcgi_buffer_size 64k;
  fastcgi_buffers 4 64k;
  fastcgi_busy_buffers_size 128k;
  fastcgi_temp_file_write_size 128k;
  # 开启 gzip 压缩
  gzip on;
  gzip_min_length  1k;
  gzip_buffers     4 16k;
  gzip_http_version 1.1;
  gzip_comp_level 2;
  gzip_types       text/plain application/x-javascript text/css application/xml;
  gzip_vary on;

  #limit_zone  crawler  $binary_remote_addr  10m;

  server
  {
    listen       80;
    server_name  www.yourdomain.com yourdomain.com;
    index index.html index.htm index.php;
```


www.TopSage.com

```
    root  /data0/htdocs;

    #limit_conn  crawler  20;

    location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
    {
      expires     30d;
    }

    location ~ .*\.(js|css)?$
    {
      expires     1h;
    }

    log_format  access  '$remote_addr - $remote_user [$time_local] "$request" '
              '$status $body_bytes_sent "$http_referer" '
              '"$http_user_agent" $http_x_forwarded_for';
    access_log  /data1/logs/access.log  access;
  }
}
```

通过上面的 nginx.conf 配置文件示例，可以看出，nginx.conf 的配置文件结构主要由以下几部分构成（见代码 3-2）。

代码 3-2

```
......
events
{
......
}

http
{
......
    server
    {
    ......
    }

    server
    {
    ......
    }
......
}
```

下一节笔者会详细介绍 Nginx 虚拟主机的配置。

www.TopSage.com

大家网 TopSage.com

3.2 Nginx 的虚拟主机配置

3.2.1 什么是虚拟主机

虚拟主机使用的是特殊的软硬件技术，它把一台运行在因特网上的服务器主机分成一台台“虚拟”的主机，每台虚拟主机都可以是一个独立的网站，可以具有独立的域名，具有完整的 Internet 服务器功能（WWW、FTP、Email 等），同一台主机上的虚拟主机之间是完全独立的。从网站访问者来看，每一台虚拟主机和一台独立的主机完全一样。

利用虚拟主机，不用为每个要运行的网站提供一台单独的 Nginx 服务器或单独运行一组 Nginx 进程。虚拟主机提供了在同一台服务器、同一组 Nginx 进程上运行多个网站的功能。

在 Nginx 配置文件（nginx.conf）中，一个最简化的虚拟主机配置如代码 3-3 所示。

代码 3-3

```
http
{
    server
    {
        listen          80 default;
        server_name      _ *;
        access_log       logs/default.access.log combined;
        location / {
            index index.html;
            root  /data0/htdocs/htdocs;
        }
    }
}
```

跟 Apache 一样，Nginx 也可以配置多种类型的虚拟主机：一是基于 IP 的虚拟主机，二是基于域名的虚拟主机，三是基于端口的虚拟主机。

3.2.2 配置基于 IP 的虚拟主机

Linux、FreeBSD 操作系统都允许添加 IP 别名。IP 别名背后的概念很简单：可以在一块物理网卡上绑定多个 IP 地址。这样就能够在使用单一网卡的同一个服务器上运行多个基于 IP 的虚拟主机。设置 IP 别名也非常容易，只须配置系统上的网络接口，让它监听额外的 IP 地址。在 Linux 系统上，可以使用标准的网络配置工具（比如 ifconfig 和 route 命令）添加 IP 别名。以下是添加 IP 别名的示例：

www.TopSage.com

先用 ifconfig 命令查看该服务器的 IP 地址。下面这台服务器有一块物理网卡设备 eth0 和本地回环设备 lo，eth0 的 IP 地址为 192.168.8.42，本地回环 lo 的 IP 地址为 127.0.0.1。

本地回环代表设备的本地虚拟接口，所以默认被看作是永远不会宕掉的接口。它的主要作用有两个：一是测试本机的网络配置，能 PING 通 127.0.0.1 说明本机的网卡和 IP 协议安装都没有问题；另一个作用是某些 SERVER/CLIENT 的应用程序在运行时须调用服务器上的资源，一般要指定 SERVER 的 IP 地址，但当该程序要在同一台机器上运行且没有别的 SERVER 时，就可以把 SERVER 的资源装在本机上，SERVER 的 IP 地址设为 127.0.0.1 也同样可以运行，如代码 3-4 所示。

代码 3-4

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:30:48:34:FF:96
          inet addr:192.168.8.42  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::230:48ff:fe34:ff96/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8397714 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2435863 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:797326757 (760.3 MiB)  TX bytes:229377030 (218.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1286 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1286 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:150843 (147.3 KiB)  TX bytes:150843 (147.3 KiB)
```

如果要在 eth0 网卡设备上添加两个 IP 别名 192.168.8.43 和 192.168.8.44，可以通过以下的 ifconfig 和 route 命令来进行：

```
/sbin/ifconfig eth0:1 192.168.8.43 broadcast 192.168.8.255 netmask 255.255.255.0 up
/sbin/route add -host 192.168.8.43 dev eth0:1

/sbin/ifconfig eth0:2 192.168.8.44 broadcast 192.168.8.255 netmask 255.255.255.0 up
/sbin/route add -host 192.168.8.44 dev eth0:2
```

这时，再执行 ifconfig 命令，就可以看到 eth0 网卡设备上绑定了两个 IP 别名，如代码 3-5 所示。

www.TopSage.com

代码 3-5

```
[root@xoyo42 ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:30:48:34:FF:96
          inet addr:192.168.8.42  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::230:48ff:fe34:ff96/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8407611 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2437149 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:798160555 (761.1 MiB)  TX bytes:229521778 (218.8 MiB)

eth0:1    Link encap:Ethernet  HWaddr 00:30:48:34:FF:96
          inet addr:192.168.8.43  Bcast:192.168.8.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

eth0:2    Link encap:Ethernet  HWaddr 00:30:48:34:FF:96
          inet addr:192.168.8.44  Bcast:192.168.8.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1290 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1290 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:151235 (147.6 KiB)  TX bytes:151235 (147.6 KiB)
```

这时候，从另外一台服务器 Ping 192.168.8.43 和 192.168.8.44 两个 IP，如果能够 Ping 通，则证明配置无误。但是，通过 ifconfig 和 route 配置的 IP 别名在服务器重启后会消失，不过可以将这两条 ifconfig 和 route 命令添加到 /etc/rc.local 文件中，让系统开机时自动运行，以下是相关命令：

```
vi /etc/rc.local
```

在文件末尾增加以下内容，然后保存即可：

```
/sbin/ifconfig eth0:1 192.168.8.43 broadcast 192.168.8.255 netmask 255.255.255.0 up
/sbin/route add -host 192.168.8.43 dev eth0:1
/sbin/ifconfig eth0:2 192.168.8.44 broadcast 192.168.8.255 netmask 255.255.255.0 up
/sbin/route add -host 192.168.8.44 dev eth0:2
```

下面开始配置基于 IP 的虚拟主机。无论是通过 IP 别名在一台服务器上配置多个 IP 地址，还是通过多块网卡在服务器上配置多个 IP 地址，在 Nginx 中都能将其配置成为基于 IP 的虚拟主机。

接下来在 Nginx 配置文件（nginx.conf）中，分别对 192.168.8.43、192.168.8.44、192.168.8.45 三个 IP 配置三个纯静态 HTML 支持的虚拟主机，如代码 3-6 所示。

www.TopSage.com

大家网 TopSage.com

代码 3-6

```
http
{
    # 第一个虚拟主机
    server
    {
        # 监听的 IP 和端口
        listen         192.168.8.43:80;
        # 主机名称
        server_name   192.168.8.43;
        # 访问日志文件存放路径
        access_log     logs/server1.access.log combined;
        location /
        {
            # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
              首页文件
            index index.html index.htm;
            # HTML 网页文件存放的目录
            root  /data0/htdocs/server1;
        }
    }
    # 第二个虚拟主机
    server
    {
        # 监听的 IP 和端口
        listen         192.168.8.44:80;
        # 主机名称
        server_name   192.168.8.44;
        # 访问日志文件存放路径
        access_log     logs/server2.access.log combined;
        location /
        {
            # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
              首页文件
            index index.html index.htm;
            # HTML 网页文件存放的目录
            root  /data0/htdocs/server2;
        }
    }
    # 第三个虚拟主机
    server
    {
        # 监听的 IP 和端口
        listen         192.168.8.45:80;
        # 主机名称
        server_name   192.168.8.45;
        # 访问日志文件存放路径
        access_log     logs/server3.access.log combined;
        location /
        {
```


www.TopSage.com

```
            # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
              首页文件
            index index.html index.htm;
            # HTML 网页文件存放的目录
            root  /data0/htdocs/server3.com;
        }
    }
}
```

从上面的配置文件中可以看出，一段 server{......}就是一个虚拟主机，如果要配置多个虚拟主机，建立多段 server{}配置即可，非常方便。监听的 IP 和端口也可以不写 IP 地址，只写端口，把它配置成“listen 80”，则表示监听该服务器上所有 IP 的 80 端口，可通过 server_name 区分不同的虚拟主机。

3.2.3 配置基于域名的虚拟主机

基于域名的虚拟主机是最常见的一种虚拟主机。只需配置你的 DNS 服务器，将每个主机名映射到正确的 IP 地址，然后配置 Nginx 服务器，令其识别不同的主机名就可以了。这种虚拟主机技术，使很多虚拟主机可以共享同一个 IP 地址，有效解决了 IP 地址不足的问题。 所以，如果没有特殊要求使你必须用一个基于 IP 的虚拟主机，最好还是使用基于域名的虚拟主机。

接下来配置基于域名的虚拟主机。在以下的示例中，配置了三个虚拟主机，第一个虚拟主机表示所有对域名 aaa.domain.com 的访问都由它来处理，第二个虚拟主机表示所有对域名 bbb.otherdomain.com 的访问都由它来处理，第三个虚拟主机表示对域名 www.domain.com、domain.com，以及除了 aaa.domain.com 之外的所有*.domain.com 二级域名的访问都由它来处理。每个虚拟主机的网页文件分别存放在了不同的目录中，每个虚拟主机使用了不同的日志文件来记录访问日志，如代码 3-7 所示。

代码 3-7

```
http
{
    # 第一个虚拟主机
    server
    {
        # 监听的端口
        listen    80;
        # 主机名称
        server_name   aaa.domain.com;
        # 访问日志文件存放路径
        access_log     logs/aaa.domain.com.access.log combined;
        location /
        {
```

 www.TopSage.com

```
        # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
          首页文件
        index index.html index.htm;
        # HTML 网页文件存放的目录
        root  /data0/htdocs/aaa.domain.com;
    }
}
# 第二个虚拟主机
server
{
    # 监听的 IP 和端口
    listen     80;
    # 主机名称
    server_name   bbb.otherdomain.com;
    # 访问日志文件存放路径
    access_log     logs/bbb.otherdomain.com.access.log combined;
    location /
    {
        # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
          首页文件
        index index.html index.htm;
        # HTML 网页文件存放的目录
        root  /data0/htdocs/bbb.otherdomain.com;
    }
}
# 第三个虚拟主机
server
{
    # 监听的 IP 和端口
    listen     80;
    # 主机名称
    server_name   www.domain.com domain.com *.domain.com;
    # 访问日志文件存放路径
    access_log     logs/bbb.domain.com.access.log combined;
    location /
    {
        # 默认首页文件，顺序从左到右，如果找不到 index.html 文件，则查找 index.htm 文件作为
          首页文件
        index index.html index.htm;
        # HTML 网页文件存放的目录
        root  /data0/htdocs/domain.com;
    }
}
```

3.3 Nginx 的日志文件配置与切割

在上一节的 Nginx 虚拟主机配置中，已经使用 access_log 进行了日志记录，这一节中，笔者将详细介绍 Nginx 访问日志文件的配置。

www.TopSage.com

大家网 TopSage.com

与 Nginx 日志相关的指令主要有两条，一条是 log_format，用来设置日志的格式，另外一条是 access_log，用来指定日志文件的存放路径、格式和缓存大小。两条指令在 Nginx 配置文件中的位置可以在 http{......}之间，也可以在虚拟主机之间，即 server{......}两个大括号之间。

3.3.1 用 log_format 指令设置日志格式

log_format 指令用来设置日志的记录格式，它的语法如下：

```
log_format name format [format ...]
```

其中 name 表示定义的格式名称，format 表示定义的格式样式。log_format 有一个默认的、无须设置的 combined 日志格式设置，相当于 Apache 的 combined 日志格式，其具体参数如下：

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

您也可以自定义一份日志的记录格式，不过要注意，log_format 指令设置的 name 名称在 Nginx 配置文件中是不能重复的。

假设将 Nginx 服务器作为 Web 服务器，位于负载均衡设备、Squid、Nginx 反向代理之后，就不能获取到客户端的真实 IP 地址了。原因是经过反向代理后，由于在客户端和 Web 服务器之间增加了中间层，因此 Web 服务器无法直接拿到客户端的 IP，通过$remote_addr 变量拿到的将是反向代理服务器的 IP 地址。但是，反向代理服务器在转发请求的 HTTP 头信息中，可以增加 X-Forwarded-For 信息，用以记录原有的客户端 IP 地址和原来客户端请求的服务器地址。

这时候，就要用 log_format 指令来设置日志格式，让日志记录 X-Forwarded-For 信息中的 IP 地址，即客户的真实 IP。例如，创建一个名为 mylogformat 的日志格式，再用$http_x_forwarded_for 变量记录用户的 X-Forwarded-For IP 地址：

```
log_format mylogformat '$http_x_forwarded_for - $remote_user [$time_local] '
                       '"$request" $status $body_bytes_sent '
                       '"$http_referer" "$http_user_agent"';
```

在日志格式样式中，变量$remote_addr 和$http_x_forwarded_for 用于记录 IP 地址；$remote_user 用于记录远程客户端用户名称；$time_local 用于记录访问时间与时区；$request 用于记录请求 URL 与 HTTP 协议；$status 用于记录请求状态，例如成功时状态为 200，页面找到时状态为 404；$body_bytes_sent 用于记录发送给客户端的文件主体内容大小；$http_referer 用于记录是从哪个页面链接访问过来的；$http_user_agent 用于记录客户端浏览器的相关信息。

采用日志格式 combined 记录的日志格式如代码 3-8 所示。

www.TopSage.com

代码 3-8

```
124.42.4.194 - - [12/Mar/2009:02:18:23 +0800] "GET / HTTP/1.1" 200 36179 "-"
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Mozilla/4.0 (compatible; MSIE
6.0; Windows NT 5.1; SV1) ; CIBA; .NET CLR 2.0.50727)"
124.42.4.194 - - [12/Mar/2009:02:18:24 +0800] "GET /attachment.php?fid=12 HTTP/1.1"
302 5 "http://blog.s135.com/" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; CIBA; .NET CLR 2.0.50727)"
124.42.4.194 - - [12/Mar/2009:02:18:24 +0800] "GET /attachment.php?fid=2 HTTP/1.1"
302 5 "http://blog.s135.com/" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; CIBA; .NET CLR 2.0.50727)"
```

3.3.2 用 access_log 指令指定日志文件存放路径

用 log_format 指令设置了日志格式之后，需要用 access_log 指令指定日志文件存放路径。access_log 指令的语法如下：

```
access_log path [format [buffer=size | off]]
```

其中 path 表示日志文件的存放路径，format 表示使用 log_format 指令设置的日志格式的名称，buffer=size 表示设置内存缓冲区的大小，例如可以设置 buffer=32k。

（1）如果不想记录日志，可以使用以下指令关闭日志记录：

```
access_log off;
```

（2）如果想使用默认的 combined 格式的日志记录，可以使用以下示例：

```
access_log /data1/logs/filename.log;
```

或者

```
access_log /data1/logs/filename.log combined;
```

（3）如果想使用自定义格式的日志记录，可以使用以下示例，其中的 mylogformat 是日志格式名称：

```
log_format mylogformat '$remote_addr - $remote_user [$time_local] "$request" '
                       '$status $body_bytes_sent "$http_referer" '
                       '"$http_user_agent" $http_x_forwarded_for';

access_log  /data1/logs/access.log  mylogformat  buffer=32k;
```

（4）在 Nginx 0.7.4 之后的版本中，access_log 指令中的日志文件路径可以包含变量，例如：

```
access_log /data1/logs/$server_name.log combined;
```

假设 server_name 指令设置的虚拟主机名称为 test.domain.com，那么 access_log 指令将把访

www.TopSage.com

问日志记录在/data1/logs/test.domain.com.log 文件中。

如果日志文件路径中含有变量，将存在以下一些限制：

（1）Nginx 进程设置的用户和组必须有对该路径创建文件的权限。假设 Nginx 的 user 指令设置的用户名和用户组都是 www，而/data1/logs/目录的用户名和用户组为 root，日志文件/data1/logs/test.domain.com.log 将无法被 Nginx 创建；

（2）缓冲将不会被使用；

（3）对于每一条日志记录，日志文件都将先打开文件，再写入日志记录，然后马上关闭。为了提高包含变量的日志文件存放路径的性能，须要使用 open_log_file_cache 指令设置经常被使用的日志文件描述符缓存。

open_log_file_cache 指令主要用来设置含有变量的日志路径的文件描述符缓存，它的语法如下：

```
open_log_file_cache max=N [inactive=time] [min_uses=N] [valid=time] | off
```

该指令默认是禁止的，等同于：

```
open_log_file_cache off;
```

open_log_file_cache 指令的各项参数说明如下：

max：设置缓存中的最大文件描述符数量。如果超过设置的最大文件描述符数量，则采用 LRU（Least Recently Used）算法清除“较不常使用的文件描述符”。LRU（Least Recently Used）算法的基本概念是：当内存缓冲区剩余的可用空间不够时，缓冲区尽可能地先保留使用者最常使用的数据，将最近未使用的数据移出内存，腾出空间来加载另外的数据。

inactive：设置一个时间，如果在设置的时间内没有使用此文件描述符，则自动删除此描述符。此参数为可选参数，默认的时间为 10 秒钟。

min_uses：在参数 inactive 指定的时间范围内，如果日志文件超过被使用的次数，则将该日志文件的描述符记入缓存。默认次数为 1。

valid：设置多长时间检查一次，看一看变量指定的日志文件路径与文件名是否仍然存在。默认时间为 60 秒。

off：禁止使用缓存。

open_log_file_cache 指令的设置示例如下：

```
open_log_file_cache max=1000 inactive=20s min_uses=2 valid=1m;
```


www.TopSage.com

大家网 TopSage.com

3.3.3 Nginx 日志文件的切割

生产环境中的服务器，由于访问日志文件增长速度非常快，日志太大会严重影响服务器效率。同时，为了方便对日志进行分析计算，须要对日志文件进行定时切割。定时切割的方式有按月切割、按天切割、按小时切割等。最常用的是按天切割。

Nginx 不支持像 Apache 一样使用 cronolog 来轮转日志，但是可以采用以下方式来实现日志文件的切割：

```
mv /data1/logs/access.log /data1/logs/20090318.log
kill -USR1 Nginx 主进程号
```

首先通过 mv 命令将日志文件重命名为/data1/logs/20090318.log，然后发送 kill -USR1 信号给 Nginx 的主进程号，让 Nginx 重新生成一个新的日志文件/data1/logs/access.log。如果 nginx.conf 配置文件中使用了“pid /usr/local/webserver/nginx/nginx.pid;”指令，指定了 pid 文件的存放路径，我们可以通过 cat 这个 pid 文件获得 Nginx 的主进程号，命令如下：

```
kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
```

如果想每天定时切割日志，还须要借助 crontab。我们可以写一个按天切割的日志，按年、月份目录存放日志的 shell 脚本：

```
vi /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

输入以下内容并保存（见代码 3-9）：

代码 3-9

```
#!/bin/bash
# 这个脚本须在每天的 00:00 运行

# Nginx 日志文件的存放路径
logs_path="/data1/logs/"

mkdir -p ${logs_path}$(date -d "yesterday" +"%Y")/$(date -d "yesterday" +"%m")/
mv ${logs_path}access.log ${logs_path}$(date -d "yesterday" +"%Y")/$(date -d
"yesterday" +"%m")/access_$(date -d "yesterday" +"%Y%m%d").log
kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
```

另外，配置 crontab 每天凌晨 00:00 定时执行这个脚本：

```
crontab -e
```

输入以下内容并保存：

www.TopSage.com

```
00 00 * * * /bin/bash  /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

这个 shell 脚本和 crontab 配置主要实现的功能为：假设今天的日期为 2009 年 5 月 19 日，Nginx 当前的日志文件为/data1/logs/access.log，2009 年 5 月 20 日 00:00 会执行 cut_nginx_log.sh 脚本，cut_nginx_log.sh 脚本首先创建一个目录/data1/logs/2009/05/，然后将/data1/logs/access.log 文件移动并重命名为/data1/logs/2009/05/access_20090519.log，再发送 kill -USR1 信号给 Nginx 主进程号，告诉 Nginx 重新生成一个/data1/logs/access.log 文件，2009 年 5 月 20 日的日志记录在这个新生成的日志文件中。而/data1/logs/2009/05/access_20090519.log 文件，就是 2009 年 5 月 19 日的日志文件。

3.4 Nginx 的压缩输出配置

gzip（GNU-ZIP）是一种压缩技术。经过 gzip 压缩后页面大小可以变为原来的 30%甚至更小。这样，用户浏览页面的时候速度会快得多。gzip 的压缩页面需要浏览器和服务器双方都支持，实际上就是服务器端压缩，传到浏览器后浏览器解压并解析。浏览器那里不需要我们担心，因为 IE、Firefox、Opera、谷歌 Chrome 等绝大多数浏览器都支持解析 gzip 过的页面。

Nginx 的压缩输出由一组 gzip 压缩指令来实现。我们从示例入手，来学习 gzip 压缩输出。gzip 压缩输出的相关指令位于 http{……}两个大括号之间：

```
gzip on;
gzip_min_length  1k;
gzip_buffers     4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css application/xml;
gzip_vary on;
```

gzip 各指令的语法和使用方法请查阅本书第 13 章第 13.11 节的 gzip 模块。

3.5 Nginx 的自动列目录配置

我们经常会看到一些开源软件的下载页面是能够自动列目录的，这一功能 Apache 可以实现，Nginx 同样可以实现，前提条件是当前目录下不存在用 index 指令设置的默认首页文件。如果须要在某一虚拟主机的 location / {……}目录控制中配置自动列目录，只须加上如下代码：

```
location / {
 autoindex on;
}
```

另外，还有两项跟自动列目录相关的指令，分别为：

www.TopSage.com

```
autoindex_exact_size [ on|off ]
```

设定索引时文件大小的单位（B、KB、MB 或 GB）

```
autoindex_localtime [ on|off ]
```

开启以本地时间来显示文件时间的功能。默认为关（GMT 时间）。

3.6 Nginx 的浏览器本地缓存设置

浏览器缓存（Browser Caching）是为了加速浏览，浏览器在用户磁盘上对最近请求过的文档进行存储，当访问者再次请求这个页面时，浏览器就可以从本地磁盘显示文档，这样就可以加速页面的阅览。缓存的方式节约了网络的资源，提高了网络的效率。

浏览器缓存可以通过 expires 指令输出 Header 头来实现，expires 指令的语法如下：

语法：expires [time|epoch|max|off]

默认值：expires off

作用域：http，server，location

用途：使用本指令可以控制 HTTP 应答中的“Expires”和“Cache-Control”的 Header 头信息（起到控制页面缓存的作用）。

可以在 time 值中使用正数或负数。“Expires”头标的值将通过当前系统时间加上您设定的 time 值来获得。

epoch 指定“Expires”的值为 1 January, 1970, 00:00:01 GMT。

max 指定“Expires”的值为 31 December 2037 23:59:59 GMT，“Cache-Control”的值为 10 年。–1 指定“Expires”的值为服务器当前时间–1s，即永远过期。

“Cache-Control”头标的值由您指定的时间来决定。

负数：Cache-Control: no-cache。

正数或零：Cache-Control: max-age = #, # 为您指定时间的秒数。

“off”表示不修改“Expires”和“Cache-Control”的值。

假设一个 HTML 页面中会引用一些 JavaScript 文件、图片文件，而这些格式的文件很少会被修改，则可以通过 expires 设置浏览器缓存。

例：对常见格式的图片、Flash 文件在浏览器本地缓存 30 天，对 js、css 文件在浏览器本地缓存 1 小时，如代码 3-10 所示。

www.TopSage.com



代码 3-10

```
location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
{
   expires 30d;
}

location ~ .*\.(js|css)?$
{
  expires 1h;
}
```


www.TopSage.com

第4章

Nginx 与 PHP（FastCGI）的安装、配置与优化

在互联网服务器架构中，我们经常可以听到 LAMP（Linux+Apache+Mysql+Perl/PHP/Python）架构。LAMP 这个特定名词最早出现在 1998 年。当时，Michael Kunze 为德国计算机杂志《c't》写作的一篇关于自由软件如何成为商业软件替代品的文章时，创建了 LAMP 这个名词，用来指代 Linux 操作系统、Apache 网络服务器、MySQL 数据库和 PHP（Perl 或 Python）脚本语言的组合（由 4 种技术开头的字母组成）。由于 IT 世界众所周知的对缩写的爱好，Michael Kunze 提出的 LAMP 这一术语很快就被市场接受。O'Reilly 和 MySQL AB 更是在英语人群中推广普及了这个术语。随之 LAMP 技术成了开源软件业的一盏真正的明灯。

虽然这些开放源代码程序本身并不是专门设计成同另外几个程序一起工作的，但由于它们都是影响较大的开源软件，拥有很多共同特点，这就导致了这些组件经常在一起使用。在过去的几年里，这些组件的兼容性不断完善，在一起应用的情形变得更加普遍。并且它们为了改善不同组件之间的协作，已经创建了某些扩展功能。目前，几乎在所有的 Linux 发布版中都默认包含了这些产品。Linux 操作系统、Apache 服务器、MySQL 数据库和 Perl、PHP 或 Python 语言，这些产品共同组成了一个强大的 Web 应用程序平台。

现在，由于 Nginx 拥有超越 Apache 的卓越性能，LAMP 架构正在逐渐被 LNMP 架构所取代。LNMP 即为我们本章详细介绍的 Linux+Nginx+Mysql+PHP 架构。本章我们将一步步学习 Nginx

www.TopSage.com

与 PHP（FastCGI）的安装、配置与优化。

提高 PHP（FastCGI），那么 FastCGI 是什么呢？

FastCGI 是语言无关的、可伸缩架构的 CGI 开放扩展，其主要行为是将 CGI 解释器进程保持在内存中并因此获得较高的性能。众所周知，CGI 解释器的反复加载是 CGI 性能低下的主要原因，如果 CGI 解释器保持在内存中并接受 FastCGI 进程管理器调度，则可以提供良好的性能、伸缩性、Fail-Over 特性等。

FastCGI 的工作原理是：

（1）FastCGI 进程管理器自身初始化，启动多个 CGI 解释器进程（多个 php-cgi 进程）并等待来自 Web Server 的连接。在本文中，采用 PHP-FPM 进程管理器启动多个 php-cgi FastCGI 进程。启动 php-cgi FastCGI 进程时，可以配置以 TCP 和 UNIX 套接字两种方式启动。

（2）当客户端请求到达 Web 服务器（Nginx）时，Web 服务器将请求采用 TCP 协议或 UNIX 套接字方式转发到 FastCGI 主进程，FastCGI 主进程选择并连接到一个 CGI 解释器（子进程）。Web 服务器将 CGI 环境变量和标准输入发送到 FastCGI 子进程 php-cgi。

（3）FastCGI 子进程完成处理后将标准输出和错误信息从同一连接返回 Web 服务器（Nginx）。当 FastCGI 子进程关闭连接时，请求便告知处理完成。FastCGI 子进程接着等待并处理来自 FastCGI 进程管理器的下一个连接。而在一般的普通 CGI 模式中，php-cgi 在此便退出了。

所以，你可以想象普通的 CGI 模式有多慢。每一个 Web 请求 PHP 都必须重新解析 php.ini、重新载入全部扩展并重新初始化全部数据结构。使用 FastCGI，所有这些都只在进程启动时发生一次。一个额外的好处是，持续数据库连接（Persistent database connection）可以工作。

PHP FastCGI 的优点：

（1）PHP 脚本运行速度更快。PHP 解释程序被载入内存而不用每次需要时从存储器读取，此举极大提升了依靠脚本运行站点的性能。

（2）须要使用的系统资源更少。由于服务器不用在每次需要时都载入 PHP 解释程序，你可以将站点的传输速度提升很多而不必增加 CPU 负担。

（3）不需要对现有的代码作任何改变。运行在 Apache+PHP 上的程序，无须修改即可适用于 PHP 的 FastCGI。

接下来，我们开始安装、搭建 LNMP（Linux+Nginx+Mysql+PHP）平台。本文中的操作系统环境为：CentOS Linux 5.3（Linux 2.6+内核），另在 RedHat AS4 上也可安装成功。

www.TopSage.com

大家网 TopSage.com

4.1 获取相关开源程序

1. 【适用 CentOS 操作系统】

利用 CentOS Linux 系统自带的 yum 命令安装、升级所需的程序库（RedHat 等其他 Linux 发行版可从安装光盘中找到这些程序库的 RPM 包，进行安装）：

```
sudo -s
LANG=C
yum -y install gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype
freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2
glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-devel e2fsprogs
e2fsprogs-devel krb5 krb5-devel libidn libidn-devel openssl openssl-devel openldap
openldap-devel nss_ldap openldap-clients openldap-servers
```

2. 【适用 RedHat 操作系统】

RedHat 等其他 Linux 发行版可从安装光盘中找到这些程序库的 RPM 包（事先可通过类似“rpm -qa | grep libjpeg”的命令查看 RPM 包是否存在，但通常“xxx-devel”不存在，须要安装）。RedHat 可以直接利用 CentOS 的 RPM 包安装，以下是 RPM 包的下载网址：

（1）RedHat AS4 & CentOS 4：

http://mirrors.163.com/centos/4/os/i386/CentOS/RPMS/

http://mirrors.163.com/centos/4/os/x86_64/CentOS/RPMS/

（2）RedHat AS5 & CentOS 5：

http://mirrors.163.com/centos/5/os/i386/CentOS/

http://mirrors.163.com/centos/5/os/x86_64/CentOS/

（3）RPM 包搜索网站：

http://rpm.pbone.net/

http://www.rpmfind.net/

（4）RedHat AS4 系统环境，通常情况下缺少包安装支持：

I. i386 系统如代码 4-1：

代码 4-1

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/libjpeg-devel-6b-33.i386.rpm
rpm -ivh libjpeg-devel-6b-33.i386.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/freetype-devel-2.1.9-1.i386.rpm
rpm -ivh freetype-devel-2.1.9-1.i386.rpm
```


www.TopSage.com

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/i386/libpng-devel-1.2.7-1.i386.rpm
rpm -ivh libpng-devel-1.2.7-1.i386.rpm
```

II. x86_64 系统如代码 4-2：

代码 4-2

```
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/libjpeg-devel-6b-33.
x86_64.rpm
rpm -ivh libjpeg-devel-6b-33.x86_64.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/freetype-devel-2.1.9-1.
x86_64.rpm
rpm -ivh freetype-devel-2.1.9-1.x86_64.rpm
wget http://blog.s135.com/soft/linux/nginx_php/rpm/x86_64/libpng-devel-1.2.7-1.
x86_64.rpm
rpm -ivh libpng-devel-1.2.7-1.x86_64.rpm
```

3. 【适用 CentOS、RedHat 及其他 Linux 操作系统】

下载程序源码包：

本文中提到的所有开源软件为截止到 2009 年 09 月 18 日的最新稳定版。

（1）从软件的官方网站下载如代码 4-3：

代码 4-3

```
mkdir -p /data0/software
cd /data0/software
wget http://sysoev.ru/nginx/nginx-0.8.15.tar.gz
wget http://www.php.net/get/php-5.2.10.tar.gz/from/this/mirror
wget http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.10-fpm-0.5.11.diff.gz
wget http://dev.mysql.com/get/Downloads/MySQL-5.1/mysql-5.1.38.tar.gz/from/
http://mysql.he.net/
wget http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.13.tar.gz
wget "http://downloads.sourceforge.net/mcrypt/libmcrypt-2.5.8.tar.gz? modtime=
1171868460&big_mirror=0"
wget "http://downloads.sourceforge.net/mcrypt/mcrypt-2.6.8.tar.gz?modtime=
1194463373&big_mirror=0"
wget http://pecl.php.net/get/memcache-2.2.5.tgz
wget "http://downloads.sourceforge.net/mhash/mhash-0.9.9.9.tar.gz?modtime=
1175740843&big_mirror=0"
wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.9.tar.gz
wget http://bart.eaccelerator.net/source/0.9.5.3/eaccelerator-0.9.5.3. tar.bz2
wget http://pecl.php.net/get/PDO_MYSQL-1.0.2.tgz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
wget http://pecl.php.net/get/imagick-2.2.2.tgz
```

（2）从 blog.s135.com 下载（比较稳定，只允许在本站，或者在 Linux/Unix 下通过 Wget、Curl 等命令下载以下软件），如代码 4-4：

www.TopSage.com

代码 4-4

```
mkdir -p /data0/software
cd /data0/software
wget http://blog.s135.com/soft/linux/nginx_php/nginx/nginx-0.8.15.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/php/php-5.2.10.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.10-fpm-0.5.11.diff.gz
wget http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.1.38.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/libiconv/libiconv-1.13.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/mcrypt/libmcrypt- 2.5.8.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/mcrypt/mcrypt-2.6.8.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/memcache/memcache-2.2.5.tgz
wget http://blog.s135.com/soft/linux/nginx_php/mhash/mhash-0.9.9.9.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-7.9.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/eaccelerator/eaccelerator-
0.9.5.3.tar.bz2
wget http://blog.s135.com/soft/linux/nginx_php/pdo/PDO_MYSQL-1.0.2.tgz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
wget http://blog.s135.com/soft/linux/nginx_php/imagick/imagick-2.2.2.tgz
```

4.2 安装 PHP 5.2.10（FastCGI 模式）

（1）编译安装 PHP 5.2.10 所需的支持库，代码如 4-5 所示：

代码 4-5

```
tar zxvf libiconv-1.13.tar.gz
cd libiconv-1.13/
./configure --prefix=/usr/local
make
make install
cd ../

tar zxvf libmcrypt-2.5.8.tar.gz
cd libmcrypt-2.5.8/
./configure
make
make install
/sbin/ldconfig
cd libltdl/
./configure --enable-ltdl-install
make
make install
cd ../../

tar zxvf mhash-0.9.9.9.tar.gz
cd mhash-0.9.9.9/
./configure
make
```


www.TopSage.com

大家网 TopSage.com

```
make install
cd ../

ln -s /usr/local/lib/libmcrypt.la /usr/lib/libmcrypt.la
ln -s /usr/local/lib/libmcrypt.so /usr/lib/libmcrypt.so
ln -s /usr/local/lib/libmcrypt.so.4 /usr/lib/libmcrypt.so.4
ln -s /usr/local/lib/libmcrypt.so.4.4.8 /usr/lib/libmcrypt.so.4.4.8
ln -s /usr/local/lib/libmhash.a /usr/lib/libmhash.a
ln -s /usr/local/lib/libmhash.la /usr/lib/libmhash.la
ln -s /usr/local/lib/libmhash.so /usr/lib/libmhash.so
ln -s /usr/local/lib/libmhash.so.2 /usr/lib/libmhash.so.2
ln -s /usr/local/lib/libmhash.so.2.0.1 /usr/lib/libmhash.so.2.0.1

tar zxvf mcrypt-2.6.8.tar.gz
cd mcrypt-2.6.8/
/sbin/ldconfig
./configure
make
make install
cd ../
```

（2）编译安装 MySQL 5.1.38 所需的代码如 4-6 所示：

代码 4-6

```
/usr/sbin/groupadd mysql
/usr/sbin/useradd -g mysql mysql
tar zxvf mysql-5.1.38.tar.gz
cd mysql-5.1.38/
./configure --prefix=/usr/local/webserver/mysql/ --enable-assembler
--with-extra-charsets=complex --enable-thread-safe-client --with-big-tables
--with-readline --with-ssl --with-embedded-server --enable-local-infile
--with-plugins=innobase
make && make install
chmod +w /usr/local/webserver/mysql
chown -R mysql:mysql /usr/local/webserver/mysql
cd ../
```

如果你想在这台服务器上运行 MySQL 数据库，则执行以下几步。如果你只是希望让 PHP 支持 MySQL 扩展库，能够连接其他服务器上的 MySQL 数据库，那么，以下几步无须执行。

1）创建 MySQL 数据库存放目录：

```
mkdir -p /data0/mysql/3306/data/
chown -R mysql:mysql /data0/mysql/
```

2）以 mysql 用户账号的身份建立数据表：

```
/usr/local/webserver/mysql/bin/mysql_install_db
--basedir=/usr/local/webserver/mysql --datadir=/data0/mysql/3306/data
--user=mysql
```

3）创建 my.cnf 配置文件：

www.TopSage.com

```
vi /data0/mysql/3306/my.cnf
```

输入代码 4-7 的内容如下：

代码 4-7

```
[client]
default-character-set = utf8
port = 3306
socket = /tmp/mysql.sock

[mysql]
prompt="(\u:blog.domain.com:)[\d]> "
no-auto-rehash

[mysqld]
#default-character-set = utf8
user = mysql
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /data0/mysql/3306/data
open_files_limit    = 10240
back_log = 600
max_connections = 3000
max_connect_errors = 6000
table_cache = 614
external-locking = FALSE
max_allowed_packet = 32M
sort_buffer_size = 2M
join_buffer_size = 2M
thread_cache_size = 300
thread_concurrency = 8
query_cache_size = 32M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default-storage-engine = MyISAM
default_table_type = MyISAM
thread_stack = 192K
transaction_isolation = READ-COMMITTED
tmp_table_size = 246M
max_heap_table_size = 246M
long_query_time = 1
log_long_format
log-bin = /data0/mysql/3306/binlog
binlog_cache_size = 4M
binlog_format = MIXED
max_binlog_cache_size = 8M
max_binlog_size = 512M
expire_logs_days = 7
key_buffer_size = 256M
read_buffer_size = 1M
```

www.TopSage.com

```
read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

skip-name-resolve
master-connect-retry = 10
slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

server-id = 1

innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 2048M
innodb_data_file_path = ibdata1:1024M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 16M
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0
[mysqldump]
quick
max_allowed_packet = 32M
```

4）创建管理 MySQL 数据库的 shell 脚本：

```
vi /data0/mysql/3306/mysql
```

输入代码 4-8 的内容（这里的用户名 admin 和密码 12345678 接下来的步骤会创建）如下：

代码 4-8

```
#!/bin/sh

mysql_port=3306
mysql_username="admin"
mysql_password="12345678"

function_start_mysql()
{
   printf "Starting MySQL...\n"
   /bin/sh /usr/local/webserver/mysql/bin/mysqld_safe --defaults-file=
/data0/mysql/${mysql_port}/my.cnf 2>&1 > /dev/null &
}

function_stop_mysql()
{
```


www.TopSage.com




```
   printf "Stoping MySQL...\n"
   /usr/local/webserver/mysql/bin/mysqladmin -u ${mysql_username}
-p${mysql_password} -S /tmp/mysql.sock shutdown
}

function_restart_mysql()
{
   printf "Restarting MySQL...\n"
   function_stop_mysql
   sleep 5
   function_start_mysql
}

function_kill_mysql()
{
   kill -9 $(ps -ef | grep 'bin/mysqld_safe' | grep ${mysql_port} | awk '{printf $2}')
   kill -9 $(ps -ef | grep 'libexec/mysqld' | grep ${mysql_port} | awk '{printf $2}')
}

if [ "$1" = "start" ]; then
   function_start_mysql
elif [ "$1" = "stop" ]; then
   function_stop_mysql
elif [ "$1" = "restart" ]; then
function_restart_mysql
elif [ "$1" = "kill" ]; then
function_kill_mysql
else
   printf "Usage: /data0/mysql/${mysql_port}/mysql {start|stop|restart|kill}\n"
fi
```

5）赋予 shell 脚本可执行权限：

```
chmod +x /data0/mysql/3306/mysql
```

6）启动 MySQL：

```
/data0/mysql/3306/mysql start
```

7）通过命令行登录管理 MySQL 服务器（提示输入密码时直接回车）：

```
/usr/local/webserver/mysql/bin/mysql -u root -p -S /tmp/mysql.sock
```

8）输入以下 SQL 语句，创建一个具有 root 权限的用户（admin）和密码（12345678）：

```
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' IDENTIFIED BY '12345678';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'127.0.0.1' IDENTIFIED BY '12345678';
```

9）（可选）停止 MySQL：

```
/data0/mysql/3306/mysql stop
```

（3）编译安装 PHP（FastCGI 模式），代码如 4-9 所示：

www.TopSage.com

代码 4-9

```
tar zxvf php-5.2.10.tar.gz
gzip -cd php-5.2.10-fpm-0.5.11.diff.gz | patch -d php-5.2.10 -p1
cd php-5.2.10/
./configure --prefix=/usr/local/webserver/php
--with-config-file-path=/usr/local/webserver/php/etc
--with-mysql=/usr/local/webserver/mysql
--with-mysqli=/usr/local/webserver/mysql/bin/mysql_config
--with-iconv-dir=/usr/local --with-freetype-dir --with-jpeg-dir --with-png-dir
--with-zlib --with-libxml-dir=/usr --enable-xml --disable-rpath
--enable-discard-path --enable-safe-mode --enable-bcmath --enable-shmop
--enable-sysvsem --enable-inline-optimization --with-curl --with-curlwrappers
--enable-mbregex --enable-fastcgi --enable-fpm --enable-force-cgi-redirect
--enable-mbstring --with-mcrypt --with-gd --enable-gd-native-ttf --with-openssl
--with-mhash --enable-pcntl --enable-sockets --with-ldap --with-ldap-sasl
--with-xmlrpc --enable-zip --enable-soap --without-pear
make ZEND_EXTRA_LIBS='-liconv'
make install
cp php.ini-dist /usr/local/webserver/php/etc/php.ini
cd ../
curl http://pear.php.net/go-pear | /usr/local/webserver/php/bin/php
```

（4）编译安装 PHP5 扩展模块，如代码 4-10 所示：

代码 4-10

```
tar zxvf memcache-2.2.5.tgz
cd memcache-2.2.5/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

tar jxvf eaccelerator-0.9.5.3.tar.bz2
cd eaccelerator-0.9.5.3/
/usr/local/webserver/php/bin/phpize
./configure --enable-eaccelerator=shared
--with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

tar zxvf PDO_MYSQL-1.0.2.tgz
cd PDO_MYSQL-1.0.2/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
--with-pdo-mysql=/usr/local/webserver/mysql
make
make install
cd ../
```


www.TopSage.com

```
tar zxvf ImageMagick.tar.gz
cd ImageMagick-6.5.1-2/
./configure
make
make install
cd ../

tar zxvf imagick-2.2.2.tgz
cd imagick-2.2.2/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../
```

（5）修改 php.ini 文件：

```
手工修改：查找/usr/local/webserver/php/etc/php.ini 中的 extension_dir = "./"
修改为 extension_dir = "/usr/local/webserver/php/lib/php/extensions/no-
debug-non-zts-20060613/"
```

并在此行后增加以下几行，然后保存，如代码 4-11 所示：

代码 4-11

```
extension = "memcache.so"
extension = "pdo_mysql.so"
extension = "imagick.so"
再查找 output_buffering = Off
修改为 output_buffering = On
```

自动修改：若嫌手工修改麻烦，可执行以下 shell 命令，自动完成对 php.ini 文件的修改，如代码 4-12 所示：

代码 4-12

```
sed -i 's#extension_dir = "./"#extension_dir =
"/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-20060613/"
\nextension = "memcache.so"\nextension = "pdo_mysql.so"\nextension =
"imagick.so"\n#' /usr/local/webserver/php/etc/php.ini
sed -i 's#output_buffering = Off#output_buffering = On#'
/usr/local/webserver/php/etc/php.ini
sed -i "s#; always_populate_raw_post_data = On#always_populate_raw_post_data = On#g"
/usr/local/webserver/php/etc/php.ini
```

（6）配置 eAccelerator 加速 PHP：

```
mkdir -p /usr/local/webserver/eaccelerator_cache
vi /usr/local/webserver/php/etc/php.ini
```

按 shift+g 键跳到配置文件的最末尾，加上以下配置信息，如代码 4-13 所示：

www.TopSage.com

大家网 TopSage.com

代码 4-13

```
[eaccelerator]
zend_extension="/usr/local/webserver/php/lib/php/extensions/
no-debug-non-zts-20060613/eaccelerator.so"
eaccelerator.shm_size="64"
eaccelerator.cache_dir="/usr/local/webserver/eaccelerator_cache"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="3600"
eaccelerator.shm_prune_period="3600"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
```

（7）创建 www 用户和组，以及供 blog.domain.com 和 www.domain.com 两个虚拟主机使用的目录，如代码 4-14 所示：

代码 4-14

```
/usr/sbin/groupadd www
/usr/sbin/useradd -g www www
mkdir -p /data0/htdocs/blog
chmod +w /data0/htdocs/blog
chown -R www:www /data0/htdocs/blog
mkdir -p /data0/htdocs/www
chmod +w /data0/htdocs/www
chown -R www:www /data0/htdocs/www
```

（8）创建 php-fpm 配置文件（php-fpm 是为 PHP 打的一个 FastCGI 管理补丁，可以平滑变更 php.ini 配置而无须重启 php-cgi）。

在/usr/local/webserver/php/etc/目录中创建 php-fpm.conf 文件：

```
rm -f /usr/local/webserver/php/etc/php-fpm.conf
vi /usr/local/webserver/php/etc/php-fpm.conf
```

输入以下内容（如果您安装 Nginx + PHP 用于程序调试，请将以下的<value name="display_errors">0</value>改为<value name="display_errors">1</value>，以便显示 PHP 错误信息，否则，Nginx 会报状态为 500 的空白错误页），如代码 4-15 所示：

代码 4-15

```
<?xml version="1.0" ?>
<configuration>

  All relative paths in this config are relative to php's install prefix
```


www.TopSage.com

```
<section name="global_options">

  Pid file
  <value name="pid_file">/usr/local/webserver/php/logs/php-fpm.pid</value>

  Error log file
  <value name="error_log">/usr/local/webserver/php/logs/php-fpm.log</value>

  Log level
  <value name="log_level">notice</value>

  When this amount of php processes exited with SIGSEGV or SIGBUS ...
  <value name="emergency_restart_threshold">10</value>

  ... in a less than this interval of time, a graceful restart will be initiated.
  Useful to work around accidental curruptions in accelerator's shared memory.
  <value name="emergency_restart_interval">1m</value>

  Time limit on waiting child's reaction on signals from master
  <value name="process_control_timeout">5s</value>

  Set to 'no' to debug fpm
  <value name="daemonize">yes</value>

</section>

<workers>

  <section name="pool">

    Name of pool. Used in logs and stats.
    <value name="name">default</value>

    Address to accept fastcgi requests on.
    Valid syntax is 'ip.ad.re.ss:port' or just 'port' or'/path/to/unix/socket'
    <value name="listen_address">127.0.0.1:9000</value>

    <value name="listen_options">

      Set listen(2) backlog
      <value name="backlog">-1</value>

      Set permissions for unix socket, if one used.
      In Linux read/write permissions must be set in order to allow connections
           from web server.
      Many BSD-derrived systems allow connections regardless of permissions.
      <value name="owner"></value>
      <value name="group"></value>
      <value name="mode">0666</value>
    </value>
```

www.TopSage.com

```
Additional php.ini defines, specific to this pool of workers.
<value name="php_defines">
  <value name="sendmail_path">/usr/sbin/sendmail -t -i</value>
  <value name="display_errors">1</value>
</value>

Unix user of processes
  <value name="user">www</value>

Unix group of processes
  <value name="group">www</value>

Process manager settings
  <value name="pm">

  Sets style of controling worker process count.
  Valid values are 'static' and 'apache-like'
  <value name="style">static</value>

  Sets the limit on the number of simultaneous requests that will be served.
  Equivalent to Apache MaxClients directive.
  Equivalent to PHP_FCGI_CHILDREN environment in original php.fcgi
  Used with any pm_style.
  <value name="max_children">128</value>

  Settings group for 'apache-like' pm style
  <value name="apache_like">

  Sets the number of server processes created on startup.
  Used only when 'apache-like' pm_style is selected
  <value name="StartServers">20</value>

  Sets the desired minimum number of idle server processes.
  Used only when 'apache-like' pm_style is selected
  <value name="MinSpareServers">5</value>

  Sets the desired maximum number of idle server processes.
  Used only when 'apache-like' pm_style is selected
  <value name="MaxSpareServers">35</value>

  </value>

</value>
The timeout (in seconds) for serving a single request after which the worker
      process will be terminated
Should be used when 'max_execution_time' ini option does not stop script
      execution for some reason
'0s' means 'off'
<value name="request_terminate_timeout">0s</value>
The timeout (in seconds) for serving of single request after which a php
      backtrace will be dumped to slow.log file
'0s' means 'off'
```


www.TopSage.com



```
    <value name="request_slowlog_timeout">0s</value>
    The log file for slow requests
    <value name="slowlog">logs/slow.log</value>
    Set open file desc rlimit
    <value name="rlimit_files">65535</value>
    Set max core size rlimit
    <value name="rlimit_core">0</value>
    Chroot to this directory at the start, absolute path
    <value name="chroot"></value>
    Chdir to this directory at the start, absolute path
    <value name="chdir"></value>
    Redirect workers' stdout and stderr into main error log.
    If not set, they will be redirected to /dev/null, according to FastCGI specs
    <value name="catch_workers_output">yes</value>
    How much requests each process should execute before respawn.
    Useful to work around memory leaks in 3rd party libraries.
    For endless request processing please specify 0
    Equivalent to PHP_FCGI_MAX_REQUESTS
    <value name="max_requests">102400</value>
    Comma separated list of ipv4 addresses of FastCGI clients that allowed to connect.
    Equivalent to FCGI_WEB_SERVER_ADDRS environment in original php.fcgi (5.2.2+)
    Makes sense only with AF_INET listening socket.
    <value name="allowed_clients">127.0.0.1</value>
    Pass environment variables like LD_LIBRARY_PATH
    All $VARIABLEs are taken from current environment
    <value name="environment">
      <value name="HOSTNAME">$HOSTNAME</value>
      <value name="PATH">/usr/local/bin:/usr/bin:/bin</value>
      <value name="TMP">/tmp</value>
      <value name="TMPDIR">/tmp</value>
      <value name="TEMP">/tmp</value>
      <value name="OSTYPE">$OSTYPE</value>
      <value name="MACHTYPE">$MACHTYPE</value>
      <value name="MALLOC_CHECK_">2</value>
    </value>
  </section>
 </workers>
</configuration>
```

（9）启动 php-cgi 进程。

监听 127.0.0.1 的 9000 端口，进程数为 200（如果服务器内存小于 3GB，可以只开启 64 个进程），用户为 www：

```
ulimit -SHn 65535
/usr/local/webserver/php/sbin/php-fpm start
```

注：/usr/local/webserver/php/sbin/php-fpm 还有其他参数，包括：start|stop|quit|restart|reload|logrotate，修改 php.ini 后不重启 php-cgi，重新加载配置文件使用 reload。

www.TopSage.com

4.3 安装 Nginx 0.8.15

（1）安装 Nginx 所需的 pcre 库：

```
tar zxvf pcre-7.9.tar.gz
cd pcre-7.9/
./configure
make && make install
cd ../
```

（2）安装 Nginx：

```
tar zxvf nginx-0.8.15.tar.gz
cd nginx-0.8.15/
./configure --user=www --group=www --prefix=/usr/local/webserver/nginx
--with-http_stub_status_module --with-http_ssl_module
make && make install
cd ../
```

（3）创建 Nginx 日志目录：

```
mkdir -p /data1/logs
chmod +w /data1/logs
chown -R www:www /data1/logs
```

（4）创建 Nginx 配置文件。

1）在/usr/local/webserver/nginx/conf/目录中创建 nginx.conf 文件：

```
rm -f /usr/local/webserver/nginx/conf/nginx.conf
vi /usr/local/webserver/nginx/conf/nginx.conf
```

输入以下内容，如代码 4-16 所示：

代码 4-16

```
user  www www;
worker_processes 8;
error_log  /data1/logs/nginx_error.log  crit;
pid        /usr/local/webserver/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;
events
{
  use epoll;
  worker_connections 65535;
}
http
{
  include       mime.types;
  default_type  application/octet-stream;
```


www.TopSage.com

```
#charset  gb2312;
server_names_hash_bucket_size 128;
client_header_buffer_size 32k;
large_client_header_buffers 4 32k;
client_max_body_size 8m;
sendfile on;
tcp_nopush    on;
keepalive_timeout 60;
tcp_nodelay on;
fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
fastcgi_buffer_size 64k;
fastcgi_buffers 4 64k;
fastcgi_busy_buffers_size 128k;
fastcgi_temp_file_write_size 128k;
gzip on;
gzip_min_length  1k;
gzip_buffers     4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types  text/plain application/x-javascript text/css application/xml;
gzip_vary on;
#limit_zone  crawler  $binary_remote_addr  10m;
server
{
  listen      80;
  server_name  blog.domain.com;
  index index.html index.htm index.php;
  root  /data0/htdocs/blog;

  #limit_conn   crawler  20;

  location ~ .*\.(php|php5)?$
  {
    #fastcgi_pass  unix:/tmp/php-cgi.sock;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
  {
    expires      30d;
  }

  location ~ .*\.(js|css)?$
  {
    expires      1h;
  }

  log_format  access  '$remote_addr - $remote_user [$time_local]"$request" '
```


www.TopSage.com

大家网 TopSage.com

```
          '$status $body_bytes_sent "$http_referer" '
          '"$http_user_agent" $http_x_forwarded_for';
   access_log  /data1/logs/access.log  access;
    }

  server
  {
   listen      80;
   server_name  www.domain.com;
   index index.html index.htm index.php;
   root  /data0/htdocs/www;

   location ~ .*\.(php|php5)?$
   {
    #fastcgi_pass  unix:/tmp/php-cgi.sock;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
   }
   log_format wwwlogs '$remote_addr - $remote_user[$time_local]"$request" '
           '$status $body_bytes_sent "$http_referer" '
           '"$http_user_agent" $http_x_forwarded_for';
   access_log  /data1/logs/wwwlogs.log  wwwlogs;
  }
  server
  {
   listen  80;
   server_name  status.blog.domain.com;
   location / {
   stub_status on;
   access_log   off;
   }
  }
}
```

2）在/usr/local/webserver/nginx/conf/目录中创建 fcgi.conf 文件：

```
vi /usr/local/webserver/nginx/conf/fcgi.conf
```

输入以下内容，如代码 4-17 所示：

代码 4-17

```
fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx;

fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
```


www.TopSage.com

```
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

（5）启动 Nginx：

```
ulimit -SHn 65535
/usr/local/webserver/nginx/sbin/nginx
```

4.4 配置开机自动启动 Nginx + PHP

用 vi 编辑器打开文件/etc/rc.local：

```
vi /etc/rc.local
```

在末尾增加以下内容：

```
ulimit -SHn 65535
/usr/local/webserver/php/sbin/php-fpm start
/usr/local/webserver/nginx/sbin/nginx
```

4.5 优化 Linux 内核参数

用 vi 编辑器打开文件/etc/sysctl.conf：

```
vi /etc/sysctl.conf
```

在末尾增加以下内容，如代码 4-18 所示：

代码 4-18

```
# Add
net.ipv4.tcp_max_syn_backlog = 65536
net.core.netdev_max_backlog =  32768
net.core.somaxconn = 32768

net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
```


www.TopSage.com

```
net.core.wmem_max = 16777216

net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2

net.ipv4.tcp_tw_recycle = 1
#net.ipv4.tcp_tw_len = 1
net.ipv4.tcp_tw_reuse = 1

net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_max_orphans = 3276800

#net.ipv4.tcp_fin_timeout = 30
#net.ipv4.tcp_keepalive_time = 120
net.ipv4.ip_local_port_range = 1024  65535
```

使配置立即生效：

```
/sbin/sysctl -p
```

4.6 在不停止 Nginx 服务的情况下平滑变更 Nginx 配置

（1）修改/usr/local/webserver/nginx/conf/nginx.conf 配置文件后，请执行以下命令检查配置文件是否正确：

```
/usr/local/webserver/nginx/sbin/nginx -t
```

如果屏幕显示以下两行信息，说明配置文件正确：

```
the configuration file /usr/local/webserver/nginx/conf/nginx.conf syntax is ok
the configuration file /usr/local/webserver/nginx/conf/nginx.conf was tested
successfully
```

（2）这时，输入以下命令查看 Nginx 主进程号：

```
ps -ef | grep "nginx: master process" | grep -v "grep" | awk -F ' ' '{print $2}'
```

屏幕显示的即为 Nginx 主进程号，例如：

```
6302
```

这时，执行以下命令即可使修改过的 Nginx 配置文件生效：

```
kill -HUP 6302
```

或者无须这么麻烦，找到 Nginx 的 pid 文件：

```
kill -HUP `cat /usr/local/webserver/nginx/nginx.pid`
```


www.TopSage.com

大家网 TopSage.com

4.7 编写每天定时切割 Nginx 日志的脚本

（1）创建脚本/usr/local/webserver/nginx/sbin/cut_nginx_log.sh：

```
vi /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

输入以下内容，如代码 4-19 所示：

代码 4-19

```
#!/bin/bash
# This script run at 00:00

# The Nginx logs path
logs_path="/usr/local/webserver/nginx/logs/"

mkdir -p ${logs_path}$(date -d "yesterday" +"%Y")/$(date -d "yesterday" +"%m")/
mv ${logs_path}access.log ${logs_path}$(date -d "yesterday" +"%Y")/$(date -d
"yesterday" +"%m")/access_$(date -d "yesterday" +"%Y%m%d").log
kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
```

（2）设置 crontab，每天凌晨 00:00 切割 nginx 访问日志：

```
crontab -e
```

输入以下内容：

```
00 00 * * * /bin/bash /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

www.TopSage.com

第 5 章

Nginx 与 JSP、ASP.NET、Perl 的安装与配置

在上一章，我们已经介绍了 Nginx 与 PHP（FastCGI）的安装、配置与优化。Nginx 除了可以与 PHP 语言配合外，还可以和其他的一些编程语言配合。本章将介绍 Nginx 与常用的 JSP（Tomcat）、ASP.NET（Mono+FastCGI）、Perl（FastCGI）的安装与配置。

5.1 Nginx 与 JSP（Tomcat）在 Linux 上的安装、配置

JSP（Java Server Pages）是由 Sun Microsystems 公司倡导、许多公司一起参与建立的一种动态网页技术标准。JSP 技术有点类似 ASP 技术，它是在传统的网页 HTML 文件（*.htm,*.html）中插入 Java 程序段（Scriptlet）和 JSP 标记（tag），从而形成 JSP 文件（*.jsp）。使用 JSP 开发的 Web 应用是跨平台的，既能在 Linux 下运行，也能在其他操作系统上运行。

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有 Sun 的参与和支持，最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现，Tomcat 5 支持最新的 Servlet 2.4 和 JSP 2.0 规范。因为 Tomcat 技术先进、性能稳定，而且免费，因此深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，它已成为目前比较流行的 Web 应用服务器。目前的最新版本是 6.0。

www.TopSage.com

JDK（Java Development Kit）是 Sun Microsystems 针对 Java 开发员的产品。自 Java 推出以来，JDK 已经成为使用最广泛的 Java SDK。JDK 是整个 Java 的核心，包括 Java 运行环境、Java 工具和 Java 基础的类库。JDK 是学好 Java 的第一步。从 SUN 的 JDK 5.0 开始，提供了泛型等非常实用的功能，其版本也不断更新，运行效率得到了很大的提高。

5.1.1　Tomcat 和 JDK 的安装

在 Linux 上，我们首先要安装 JDK。JDK 1.6 可以在以下网址下载：

```
http://java.sun.com/javase/downloads/widget/jdk6.jsp
```

下载完成后，修改 jdk-6u17-linux-x64.bin 的文件属性为可执行，然后执行该程序安装 JDK：

```
chmod +x jdk-6u17-linux-x64.bin
./jdk-6u17-linux-x64.bin
```

按空格键看完协议，当出现提示“Do you agree to the above license terms? [yes or no]”时，输入“yes”。安装完成后，执行以下语句：

```
mv jdk1.6.0_17 /usr/local/jdk
vi /etc/profile
```

在文件末尾增加以下内容：

```
JAVA_HOME="/usr/local/jdk"
CLASS_PATH="$JAVA_HOME/lib:$JAVA_HOME/jre/lib"
PATH=".:$PATH:$JAVA_HOME/bin"
CATALINA_HOME="/usr/local/tomcat"
export JAVA_HOME CATALINA_HOME
```

保存并退出 vi 后，执行以下命令使配置生效：

```
source /etc/profile
```

安装完成 JDK 之后，按照以下步骤安装 Tomcat 二进制版本：

```
wget http://apache.freelamp.com/tomcat/tomcat-6/v6.0.20/bin/apache-tomcat-
6.0.20.tar.gz
tar zxvf apache-tomcat-6.0.20.tar.gz
mv apache-tomcat-6.0.20 /usr/local/tomcat
cp -rf /usr/local/tomcat/webapps/* /data0/htdocs/www/
vi /usr/local/tomcat/conf/server.xml
```

查找 appBase="webapps"，修改为 appBase="/data0/htdocs/www"，其中/data0/htdocs/www 即为您的网页根目录。

安装完成后，启动 Tomcat，默认监听的是 8080 端口：

```
/usr/local/tomcat/bin/startup.sh
```

停止 Tomcat 可以使用以下命令：

www.TopSage.com

```
/usr/local/tomcat/bin/shutdown.sh
```

5.1.2 Nginx 与 Tomcat 的配置

nginx.conf 配置文件内容如代码 5-1 所示。在配置文件中，静态 HTML 网页、图片、JS、CSS、Flash 等使用 Nginx 来处理，以便得到更快的速度，文件扩展名为.jsp、.do 的请求，由 Nginx 反向代理 Tomcat HTTP 服务器来处理：

代码 5-1

```
user  www www;
worker_processes 8;
error_log  /usr/local/webserver/nginx/logs/nginx_error.log  crit;
pid     /usr/local/webserver/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;
events
{
  use epoll;
  worker_connections 65535;
}
http
{
  include      mime.types;
  default_type  application/octet-stream;
  charset  utf-8;
  server_names_hash_bucket_size 128;
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;
  client_max_body_size 300m;
  sendfile on;
  tcp_nopush     on;
  keepalive_timeout 60;
  tcp_nodelay on;
  client_body_buffer_size  512k;
  proxy_connect_timeout    5;
  proxy_read_timeout      60;
  proxy_send_timeout      5;
  proxy_buffer_size       16k;
  proxy_buffers          4 64k;
  proxy_busy_buffers_size 128k;
  proxy_temp_file_write_size 128k;
  gzip on;
  gzip_min_length  1k;
  gzip_buffers     4 16k;
  gzip_http_version 1.1;
  gzip_comp_level 2;
  gzip_types text/plain application/x-javascript text/css application/xml;
  gzip_vary on;
  upstream tomcat_server {
```

```
  server  127.0.0.1:8080;
 }
 server
 {
  listen     80;
  server_name  www.yourdomain.com;
  index index.html index.htm index.jsp default.jsp index.do default.do;
  root  /data0/htdocs/www;
  if (-d $request_filename)
  {
     rewrite ^/(.*)([^/])$ http://$host/$1$2/ permanent;
  }
  location ~ \.(jsp|jspx|do)?$ {
     proxy_set_header Host  $host;
     proxy_set_header X-Forwarded-For  $remote_addr;
     proxy_pass http://tomcat_server;
  }
  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
  {
     expires    30d;
  }
  location ~ .*\.(js|css)?$
  {
     expires    1h;
  }
  access_log  off;
  }
}
```

启动 Nginx：

```
/usr/local/webserver/nginx/sbin/nginx
```

如果 Nginx 处于运行状态，也可以使用 nginx -t 检查 nginx.conf 配置文件无错误后，使用“kill -HUP Nginx 主进程号”来平滑重启 Nginx。

Nginx 启动后，可以访问以下 URL 中的 JSP 示例程序，检查 JSP 程序能否正常运行：

```
http://www.yourdomain.com/examples/jsp/
```

5.2 Nginx 与 ASP.NET（Mono+FastCGI）在 Linux 上的安装、配置

Mono 是一个由 Novell 公司（先前是 Ximian）主持的项目。该项目的目标是创建一系列符合标准 ECMA（Ecma-334 和 Ecma-335）的.NET 工具，包括 C #编译器和共同语言（CL 即 Common Language）执行平台（Platform）。与微软的.NET 不同，Mono 项目不仅可以运行于 Windows 系统内，还可以运行于 Linux、FreeBSD、Unix、Mac OS X 和 Solaris 操作系统内，不过，部分 Windows 上的 ASP.NET 程序迁移到 Linux+Mono 平台时须要做一些移植、修改。

www.TopSage.com



Mono 项目的官方网站为 *http://www.mono-project.com/*，您可以通过该网站了解 Mono 的更多信息。

5.2.1 Mono 的安装

在 CentOS 5 Linux（注：目前只兼容 32 位系统）上，按照代码 5-2 中的步骤安装 mono：

代码 5-2

```
yum groupinstall "Development Tools"
yum install httpd build-essential gcc bzip bison pkgconfig glib-devel glib2-devel
httpd-devel libpng-devel libX11-devel freetype fontconfig pango-devel ruby ruby-rdoc
gtkhtml38-devel wget

wget http://ftp.novell.com/pub/mono/sources/mono/mono-2.6.1.tar.bz2
tar jxvf mono-2.6.1.tar.bz2
cd mono-2.6.1/
./configure --prefix=/usr
make
make install
cd ../
```

从 SVN 版本库安装 fastcgi-mono-server，如代码 5-3 所示：

代码 5-3

```
export PKG_CONFIG_PATH=/usr/lib/pkgconfig/:/usr/lib/
yum install subversion
svn co http://mono-soc-2007.googlecode.com/svn/trunk/brian/FastCgi/
fastcgi-mono-server
cd fastcgi-mono-server/
./autogen.sh
make
make install
cd ../
```

以 FastCGI 方式启动 fastcgi-mono-server2，监听本机的 9001 端口，网页根目录为/data0/htdocs/www/：

```
nohup /bin/sh /usr/local/bin/fastcgi-mono-server2 /socket=tcp:9001
/root=/data0/htdocs/www/ 2>&1 > /dev/null &
```

5.2.2 Nginx 与 ASP.NET（Mono+FastCGI）的配置

nginx.conf 配置文件内容如代码 5-4 所示。在配置文件中，静态 HTML 网页、图片、JS、CSS、Flash 等使用 Nginx 来处理，以便得到更快的速度，文件扩展名为.aspx、.asmx、.ashx、.asax、.ascx、.soap、.rem、.axd、.cs、.config、.dll 的请求，由 Nginx 交给 fastcgi-mono-server2 进程去处理：

www.TopSage.com

代码 5-4

```
user  www www;

worker_processes 8;

error_log  /usr/local/webserver/nginx/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;

events
{
  use epoll;
  worker_connections 65535;
}

http
{
  include       mime.types;
  default_type  application/octet-stream;

  charset  utf-8;

  server_names_hash_bucket_size 128;
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;
  client_max_body_size 8m;

  sendfile on;
  tcp_nopush     on;

  keepalive_timeout 60;

  tcp_nodelay on;

  fastcgi_connect_timeout 300;
  fastcgi_send_timeout 300;
  fastcgi_read_timeout 300;
  fastcgi_buffer_size 64k;
  fastcgi_buffers 4 64k;
  fastcgi_busy_buffers_size 128k;
  fastcgi_temp_file_write_size 128k;

  gzip on;
  gzip_min_length  1k;
  gzip_buffers     4 16k;
  gzip_http_version 1.1;
  gzip_comp_level 2;
```


www.TopSage.com

```
 gzip_types text/plain application/x-javascript text/css application/xml;
 gzip_vary on;

 server
 {
  listen      80;
  server_name www.yourdomain.com;
  index index.html index.htm index.aspx default.aspx;
  root  /data0/htdocs/www;

  location ~ \.(aspx|asmx|ashx|asax|ascx|soap|rem|axd|cs|config|dll)?$ {
     fastcgi_pass   127.0.0.1:9001;
     fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
     include        fastcgi_params;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
  {
     expires     30d;
  }

  location ~ .*\.(js|css)?$
  {
     expires     1h;
  }

  access_log  off;
  }
}
```

启动 Nginx：

```
/usr/local/webserver/nginx/sbin/nginx
```

如果 Nginx 处于运行状态，也可以使用 nginx -t 检查 nginx.conf 配置文件无错误后，使用“kill -HUP Nginx 主进程号”来平滑重启 Nginx。

Nginx 启动后，我们可以在/data0/htdocs/www/目录下下载一个名为 info.aspx 的 ASP.NET 探针文件，以检查 ASP.NET 程序能否正常运行：

```
cd /data0/htdocs/www/
wget http://aspnetsysinfo.googlecode.com/files/aspnetsysinfo-revision_23.zip
unzip aspnetsysinfo-revision_23.zip
```

通过浏览器访问 *http://www..yourdomain.com/info.aspx*，如果一切正常，则显示的内容如表 5-1 所示：

大家网 TopSage.com

表 5-1 访问 ASP.NET 探针显示的部分内容

System Information

Name	Value
Server Name	localhost.localdomain
Server IP	192.168.1.2
Server Domain	www.yourdomain.com
Server Port	80
Web Server Version	nginx/0.7.27
Virtual Request Path	/info.aspx
Physical Request Path	/data0/htdocs/www/info.aspx
Virtual Application Root Path	/
Physical Application Root Path	/data0/htdocs/www/
Operating System	Linux version 2.6.18-92.el5PAE (mockbuild@builder16.centos.org) (gcc version 4.1.2 20071124 (Red Hat 4.1.2-42)) #1 SMP Tue Jun 10 19:22:41 EDT 2008 -- Unix 2.6.18.92
Operating System Installation Directory	
.Net Version	2.0.50727.1433
.Net Language	Chinese (China)
Server Current Time	10-1-10 上 12 时 56 分 18 秒
System Uptime	-16.04:25:09.1510000
Script Timeout	00:01:50

Processor Information

Name	Value
Processor 0	Intel(R) Xeon(TM) CPU 2.80GHz - 2793.466 MHz
Processor 1	Intel(R) Xeon(TM) CPU 2.80GHz - 2793.466 MHz
Processor 2	Intel(R) Xeon(TM) CPU 2.80GHz - 2793.466 MHz
Processor 3	Intel(R) Xeon(TM) CPU 2.80GHz - 2793.466 MHz

Memory Information

Name	Value
Current Working Set	0 Bytes
Physical Memory Size	4152208 kB
Physical Free Memory Size	151736 kB
Swap Total Size	8385848 kB
Swap Free Size	7679308 kB

 www.TopSage.com

5.3 Nginx 与 Perl（FastCGI）在 Linux 上的安装、配置

Perl 是一种脚本语言，最初的设计者为拉里·沃尔（Larry Wall），Perl 语言于 1987 年 12 月 18 日发表。Perl 吸取了 C、sed、awk、shell scripting 及很多其他程序语言的特性。与脚本语言一样，Perl 不需要编译器和链接器来运行代码，要做的只是写出程序并告诉 Perl 来运行而已。这意味着 Perl 对于小的编程问题的快速解决方案和为大型事件创建原型来测试潜在的解决方案是十分理想的。Perl 的解释程序是开放源码的免费软件，使用 Perl 不必担心费用。Perl 能在绝大多数操作系统运行，可以方便地向不同操作系统迁移。

Perl 可以采用 FastCGI 方式与 Nginx 配合使用。

5.3.1 Perl（FastCGI）的安装

在 CentOS 5 Linux 上，我们按照以下步骤安装 Perl（FastCGI）：

```
yum install perl*
perl -MCPAN -e 'install FCGI'
perl -MCPAN -e 'install FCGI::ProcManager'
vi /usr/local/bin/cgiwrap-fcgi.pl
```

然后输入代码 5-5 所示的内容：

代码 5-5

```
#!/usr/bin/perl
use FCGI;
use Socket;
use FCGI::ProcManager;
sub shutdown { FCGI::CloseSocket($socket); exit; }
sub restart  { FCGI::CloseSocket($socket); &main; }
use sigtrap 'handler', \&shutdown, 'normal-signals';
use sigtrap 'handler', \&restart,  'HUP';
require 'syscall.ph';
use POSIX qw(setsid);

END()   { }
BEGIN() { }
{
  no warnings;
  *CORE::GLOBAL::exit = sub { die "fakeexit\nrc=" . shift() . "\n"; };
};

eval q{exit};
if ($@) {
  exit unless $@ =~ /^fakeexit/;
}
```


www.TopSage.com

```
&main;

sub daemonize() {
  chdir '/' or die "Can't chdir to /: $!";
  defined( my $pid = fork ) or die "Can't fork: $!";
  exit if $pid;
  setsid() or die "Can't start a new session: $!";
  umask 0;
}

sub main {
  $proc_manager = FCGI::ProcManager->new( {n_processes => 32} );#FastCGI 进程数
  $socket = FCGI::OpenSocket("127.0.0.1:9002", 10 ); #use IP sockets
  #$socket = FCGI::OpenSocket( "/opt/nginx/fcgi/cgi.sock", 10 )
  ; #use UNIX sockets - user running this script must have w access to the 'nginx'
    folder!!
  $request =
  FCGI::Request( \*STDIN, \*STDOUT, \*STDERR, \%req_params, $socket,
  &FCGI::FAIL_ACCEPT_ON_INTR );
  $proc_manager->pm_manage();
  if ($request) { request_loop() }
  FCGI::CloseSocket($socket);
}

sub request_loop {
  while ( $request->Accept() >= 0 ) {
    $proc_manager->pm_pre_dispatch();

    #processing any STDIN input from WebServer (for CGI-POST actions)
    $stdin_passthrough = '';
    { no warnings; $req_len = 0 + $req_params{'CONTENT_LENGTH'}; };
    if ( ( $req_params{'REQUEST_METHOD'} eq 'POST' ) && ( $req_len != 0 ) ) {
      my $bytes_read = 0;
      while ( $bytes_read < $req_len ) {
        my $data = '';
        my $bytes = read( STDIN, $data, ( $req_len - $bytes_read ) );
        last if ( $bytes == 0 || !defined($bytes) );
        $stdin_passthrough .= $data;
        $bytes_read += $bytes;
      }
    }

    #running the cgi app
    if (
      ( -x $req_params{SCRIPT_FILENAME} ) &&   #can I execute this?
      ( -s $req_params{SCRIPT_FILENAME} ) &&   #Is this file empty?
      ( -r $req_params{SCRIPT_FILENAME} )      #can I read this file?
    ) {
      pipe( CHILD_RD,   PARENT_WR );
      pipe( PARENT_ERR, CHILD_ERR );
      my $pid = open( CHILD_O, "-|" );
      unless ( defined($pid) ) {
        print("Content-type: text/plain\r\n\r\n");
```


www.TopSage.com

大家网 TopSage.com

```
    print "Error: CGI app returned no output - Executing
      $req_params{SCRIPT_FILENAME} failed !\n";
    next;
  }
  $oldfh = select(PARENT_ERR);
  $|     = 1;
  select(CHILD_O);
  $| = 1;
  select($oldfh);
  if ( $pid > 0 ) {
    close(CHILD_RD);
    close(CHILD_ERR);
    print PARENT_WR $stdin_passthrough;
    close(PARENT_WR);
    $rin = $rout = $ein = $eout = '';
    vec( $rin, fileno(CHILD_O),    1 ) = 1;
    vec( $rin, fileno(PARENT_ERR), 1 ) = 1;
    $ein    = $rin;
    $nfound = 0;

    while ( $nfound = select( $rout = $rin, undef, $ein = $eout, 10 ) ) {
      die "$!" unless $nfound != -1;
      $r1 = vec( $rout, fileno(PARENT_ERR), 1 ) == 1;
      $r2 = vec( $rout, fileno(CHILD_O),    1 ) == 1;
      $e1 = vec( $eout, fileno(PARENT_ERR), 1 ) == 1;
      $e2 = vec( $eout, fileno(CHILD_O),    1 ) == 1;

      if ($r1) {
        while ( $bytes = read( PARENT_ERR, $errbytes, 4096 ) ) {
          print STDERR $errbytes;
        }
        if ($!) {
          $err = $!;
          die $!;
          vec( $rin, fileno(PARENT_ERR), 1 ) = 0
          unless ( $err == EINTR or $err == EAGAIN );
        }
      }
      if ($r2) {
        while ( $bytes = read( CHILD_O, $s, 4096 ) ) {
          print $s;
        }
        if ( !defined($bytes) ) {
          $err = $!;
          die $!;
          vec( $rin, fileno(CHILD_O), 1 ) = 0
          unless ( $err == EINTR or $err == EAGAIN );
        }
      }
      last if ( $e1 || $e2 );
    }
    close CHILD_RD;
    close PARENT_ERR;
```


www.TopSage.com

```
      waitpid( $pid, 0 );
    } else {
      foreach $key ( keys %req_params ) {
        $ENV{$key} = $req_params{$key};
      }

      # cd to the script's local directory
      if ( $req_params{SCRIPT_FILENAME} =~ /^(.*)\/[^\/] +$/ ) {
        chdir $1;
      }
      close(PARENT_WR);
      #close(PARENT_ERR);
      close(STDIN);
      close(STDERR);

      #fcntl(CHILD_RD, F_DUPFD, 0);
      syscall( &SYS_dup2, fileno(CHILD_RD),  0 );
      syscall( &SYS_dup2, fileno(CHILD_ERR), 2 );

      #open(STDIN, "<&CHILD_RD");
      exec( $req_params{SCRIPT_FILENAME} );
      die("exec failed");
    }
  } else {
    print("Content-type: text/plain\r\n\r\n");
    print "Error: No such CGI app - $req_params{SCRIPT_FILENAME} may not exist or
      is not executable by this process.\n";
  }
 }
}
```

赋予脚本可执行权限并启动 32 个 Perl-FastCGI 进程。FastCGI 进程启动后，将监听本机的 9002 端口。

```
chmod +x /usr/local/bin/cgiwrap-fcgi.pl
cgiwrap-fcgi.pl 2>&1 > /dev/null &
```

5.3.2 Nginx 与 Perl（FastCGI）的配置

nginx.conf 配置文件内容如代码 5-6 所示。在配置文件中，静态 HTML 网页、图片、JS、CSS、Flash 等使用 Nginx 来处理，以便得到更快的速度，文件扩展名为.perl、.pl、.cgi 的请求，由 Nginx 交给 Perl（FastCGI）进程去处理：

代码 5-6

```
user  www www;

worker_processes 8;
```


www.TopSage.com

```
error_log  /usr/local/webserver/nginx/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;

events
{
  use epoll;
  worker_connections 65535;
}

http
{
  include       mime.types;
  default_type  application/octet-stream;

  charset  utf-8;

  server_names_hash_bucket_size 128;
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;
  client_max_body_size 8m;

  sendfile on;
  tcp_nopush     on;

  keepalive_timeout 60;

  tcp_nodelay on;

  fastcgi_connect_timeout 300;
  fastcgi_send_timeout 300;
  fastcgi_read_timeout 300;
  fastcgi_buffer_size 64k;
  fastcgi_buffers 4 64k;
  fastcgi_busy_buffers_size 128k;
  fastcgi_temp_file_write_size 128k;

  gzip on;
  gzip_min_length  1k;
  gzip_buffers     4 16k;
  gzip_http_version 1.1;
  gzip_comp_level 2;
  gzip_types text/plain application/x-javascript text/css application/xml;
  gzip_vary on;

  server
  {
    listen       80;
    server_name  www.yourdomain.com;
```


www.TopSage.com

大家网 TopSage.com

```
   index index.html index.htm index.pl index.perl index.cgi;
   root  /data0/htdocs/www;

   location ~ \.(pl|perl|cgi)?$ {
    fastcgi_pass   127.0.0.1:9002;
    fastcgi_index  index.cgi;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    include        fastcgi_params;
   }

   location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
   {
      expires     30d;
   }

   location ~ .*\.(js|css)?$
   {
      expires     1h;
   }

   access_log  off;
   }
}
```

启动 Nginx：

```
/usr/local/webserver/nginx/sbin/nginx
```

如果 Nginx 处于运行状态，也可以使用 nginx -t 检查 nginx.conf 配置文件无错误后，使用“kill -HUP Nginx 主进程号”来平滑重启 Nginx。

Nginx 启动后，可以在/data0/htdocs/www/目录下创建一个名为 test.cgi 的 Perl 测试文件，来检查 Perl 程序能否正常运行：

```
cd /data0/htdocs/www/
vi test.cgi
```

输入以下内容：

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>Hello, world.</body></html>";
```

然后赋予 test.cgi 文件可执行权限：

```
chmod -R 777 test.cgi
```

通过浏览器访问 *http://www..yourdomain.com/test.cgi*，如果一切正常，显示的内容如下：

```
Hello, world.
```


www.TopSage.com

第 6 章

Nginx HTTP 负载均衡和反向代理的配置与优化

6.1 什么是负载均衡和反向代理

随着网站访问量的快速增长，单台服务器已经无法承担大量用户的并发访问，必须采用多台服务器协同工作，以提高计算机系统的处理能力和计算强度，满足当前业务量的需求。而如何在完成同样功能的多个网络设备之间实现合理的业务量分配，使之不会出现一台设备过忙、而其他的设备却没有充分使用的情况。要解决这一问题，可以采用负载均衡的方法。

6.1.1 负载均衡

负载均衡是由多台服务器以对称的方式组成一个服务器集合，每台服务器都具有等价的地位，都可以单独对外提供服务而无须其他服务器的辅助。通过某种负载分担技术，将外部发送来的请求均匀分配到对称结构中的某一台服务器上，而接收到请求的服务器独立地回应客户的请求。均衡负载能够平均分配客户请求到服务器阵列，藉此快速获取重要数据，解决大量并发访问服务问题。这种群集技术可以用最少的投资获得接近于大型主机的性能。

6.1.2 反向代理

反向代理（Reverse Proxy）是指以代理服务器来接受 Internet 上的连接请求，然后将请求转

www.TopSage.com

发给内部网络上的服务器，并将从服务器上得到的结果返回给 Internet 上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

通常的代理服务器，只用于代理内部网络对 Internet 的连接请求，客户机必须指定代理服务器，并将本来要直接发送到 Web 服务器上的 http 请求发送到代理服务器中。由于外部网络上的主机并不会配置并使用这个代理服务器，普通代理服务器也被设计为在 Internet 上搜寻多个不确定的服务器，而不是针对 Internet 上多个客户机的请求访问某一个固定的服务器，因此普通的 Web 代理服务器不支持外部对内部网络的访问请求。当一个代理服务器能够代理外部网络上的主机访问内部网络时，这种代理服务的方式称为反向代理服务。此时代理服务器对外就表现为一个 Web 服务器，外部网络就可以简单把它当作一个标准的 Web 服务器而不需要特定的配置。不同之处在于，这个服务器没有保存任何网页的真实数据，所有的静态网页或 CGI 程序，都保存在内部的 Web 服务器上。因此对反向代理服务器的攻击并不会使网页信息遭到破坏，这样就增强了 Web 服务器的安全性。

反向代理方式和包过滤方式或普通代理方式并无冲突，因此可以在防火墙设备中同时使用这两种方式，其中反向代理用于外部网络访问内部网络时使用，正向代理或包过滤方式用于拒绝其他外部访问方式并提供内部网络对外部网络的访问能力。因此可以结合这些方式提供最佳的安全访问方式。

6.2 常见的 Web 负载均衡方法

Web 负载均衡的方法有很多，下面介绍几种常见的负载均衡方式。

6.2.1 用户手动选择方式

这是一种较为古老的方式，通过在主站首页入口提供不同线路、不同服务器链接的方式，来实现负载均衡。这种方式在一些提供下载业务的网站中比较常见，例如：华军软件园，如图 6-1 所示。



图 6-1 华军软件园的负载均衡方式

www.TopSage.com

大家网 TopSage.com

6.2.2 DNS 轮询方式

大多域名注册商都支持对同一主机名添加多条 A 记录，这就是 DNS 轮询，DNS 服务器将解析请求按照 A 记录的顺序，随机分配到不同的 IP 上，这样就完成了简单的负载均衡。

DNS 轮询的成本非常低，在一些不重要的服务上，被经常使用。

图 6-2 为全球第二大域名注册商——enom 的域名 Web 管理界面，这里我们为 api.bz 域名添加一个二级域名 ntp.api.bz（主机名为 ntp），并为该域名添加多条 A 记录，让其 DNS 轮询 114.80.81.72、218.75.4.130、61.129.66.79、61.153.197.226、114.80.81.1、114.80.81.69 7 个 IP，用 7 台服务器来做负载均衡。

☐	ntp	A (Address)	114.80.81.69	No Options
☐	ntp	A (Address)	114.80.81.72	No Options
☐	ntp	A (Address)	218.75.4.130	No Options
☐	ntp	A (Address)	61.129.66.79	No Options
☐	ntp	A (Address)	114.80.81.1	No Options
☐	ntp	A (Address)	61.153.197.226	No Options
☐	mail	URL Redirect	http://mail.google.com/a/api.bz	No Options
☐		A (Address)		No Options

new row | delete checked | add SRV or SPF record | park domain

Edit MX Records reset save

图 6-2 DNS 服务器的 Web 管理界面

添加完成后，我们用 Linux 下的 dig 命令查看 ntp.api.bz 域名的域名解析情况，如代码 6-1 所示：

代码 6-1

```
[root@localhost ~]# dig ntp.api.bz

; <<>> DiG 9.3.4-P1 <<>> ntp.api.bz
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50025
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 5, ADDITIONAL: 4

;; QUESTION SECTION:
;ntp.api.bz.                    IN      A

;; ANSWER SECTION:
ntp.api.bz.             1792    IN      A       114.80.81.72
ntp.api.bz.             1792    IN      A       218.75.4.130
ntp.api.bz.             1792    IN      A       61.129.66.79
```

```
ntp.api.bz.           1792    IN      A       61.153.197.226
ntp.api.bz.           1792    IN      A       114.80.81.1
ntp.api.bz.           1792    IN      A       114.80.81.69

;; AUTHORITY SECTION:
api.bz.              3592    IN      NS      dns4.name-services.com.
api.bz.              3592    IN      NS      dns1.name-services.com.
api.bz.              3592    IN      NS      dns2.name-services.com.
api.bz.              3592    IN      NS      dns3.name-services.com.
api.bz.              3592'   IN      NS      dns5.name-services.com.

;; ADDITIONAL SECTION:
dns1.name-services.com. 45929   IN      A       98.124.192.1
dns2.name-services.com. 4665    IN      A       98.124.197.1
dns3.name-services.com. 38315   IN      A       98.124.193.1
dns5.name-services.com. 45929   IN      A       98.124.196.1

;; Query time: 1 msec
;; SERVER: 219.141.136.10#53(219.141.136.10)
;; WHEN: Sun Aug 16 17:34:43 2009
;; MSG SIZE  rcvd: 300
```

我们可以发现，域名解析正常，A 记录有七个 IP。每次访问 ntp.api.nz 域名会被随机解析到其中一个 IP 上，从而实现负载均衡的目的。

虽然 DNS 轮询成本低廉，但是，DNS 负载均衡存在两个明显的缺点。

1. 可靠性低

假设一个域名 DNS 轮询多台服务器，如果其中的一台服务器发生故障，那么所有的访问该服务器的请求将不会有所回应，这是任何人都不愿意看到的。即使从 DNS 中去掉该服务器的 IP，但在 Internet 上，各地区电信、网通等宽带接入商将众多的 DNS 存放在缓存中，以节省访问时间，DNS 记录全部生效需要几个小时，甚至更久。所以，尽管 DNS 轮流在一定程度上解决了负载均衡问题，但是却存在可靠性不高的缺点。

2. 负载分配不均衡

DNS 负载均衡采用的是简单的轮询负载算法，不能区分服务器的差异，不能反映服务器的当前运行状态，不能做到为性能较好的服务器多分配请求，甚至会出现客户请求集中在某一台服务器上的情况。

DNS 服务器是按照一定的层次结构组织的，本地 DNS 服务器会缓冲已解析的域名到 IP 地址的映射，这会导致使用该 DNS 服务器的用户在一段时间内访问的是同一台 Web 服务器，导致 Web 服务器间的负载不均衡。

此外，用户本地计算机也会缓存已解析的域名到 IP 地址的映射。当多个用户计算机都缓存

www.TopSage.com

了某域名到 IP 地址的映射时，而这些用户又继续访问该域名下的网页，这时也会导致不同 Web 服务器间的负载分配不均衡。

负载不均衡可能导致的后果有：某几台服务器负荷很低，而另几台服务器负荷很高、处理缓慢；配置高的服务器分配到的请求少，而配置低的服务器分配到的请求多。

因此，DNS 轮询方式仅适用于一些可靠性要求不高的服务器集群，例如：图片服务器集群、纯静态网页服务器集群等。

6.2.3 四/七层负载均衡设备

由于 DNS 轮询的缺点，一些对可靠性要求较高的服务器集群，则通过采用四/七层负载均衡设备来实现服务器的负载均衡。

世界上第一个网络体系结构由 IBM 公司提出（1974 年，名为 SNA），以后其他公司也相继提出自己的网络体系结构如：Digital 公司的 DNA，美国国防部的 TCP/IP 等，多种网络体系结构并存，其结果是若采用 IBM 的结构，只能选用 IBM 的产品，只能与同种结构的网络互联。为了促进计算机网络的发展，国际标准化组织 ISO 于 1977 年成立了一个委员会，在现有网络的基础上，提出了不基于具体机型、操作系统或公司的网络体系结构，称为开放系统互联模型（OSI，open system interconnection）。

这个模型把网络通信的工作分为七层（可参见图 6-3）。一至四层被认为是低层，这些层与数据移动密切相关。五至七层是高层，包含应用程序级的数据。每一层负责一项具体的工作，然后把数据传送到下一层。由低到高具体分为：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。

OSI 模型的最低层或第一层：物理层

物理层包括物理连网媒介，实际上就是布线、光纤、网卡和其他用来把两台网络通信设备连接在一起的设施。它规定了激活、维持、关闭通信端点之间的机械特性、电气特性、功能特性及过程特性。虽然物理层不提供纠错服务，但它能够设定数据传输速率并监测数据出错率。

OSI 模型的第二层：数据链路层

数据链路层的主要作用是控制网络层与物理层之间的通信。它保证了数据在不可靠的物理线路上进行可靠的传递。它把从网络层接收到的数据分割成特定的可被物理层传输的帧，保证了传输的可靠性。它的主要作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。它是独立于网络层和物理层的，工作时无须关心计算机是否正在运行软件还是其他操作。

数据链路层协议的主要内容包括：SDLC、HDLC、PPP、STP、帧中继等。

www.TopSage.com

大家网 TopSage.com

OSI 模型的第三层：网络层

很多用户经常混淆第二层和第三层的界限，简单来说，如果你在谈论一个与 IP 地址、路由协议或地址解析协议（ARP）相关的问题，那么这就是第三层的问题。

网络层负责对子网间的数据包进行路由选择，它通过综合考虑发送优先权、网络拥塞程度、服务质量及可选路由的花费来决定一个网络中两个节点的最佳路径。另外，它还可以实现拥塞控制、网际互联等功能。

网络层协议的主要内容包括：IP、IPX、RIP、OSPF 等。

OSI 模型的第四层：传输层

传输层是 OSI 模型中最重要的一层，它是两台计算机经过网络进行数据通信时，第一个端到端的层次，起到缓冲作用。当网络层的服务质量不能满足要求时，它将提高服务，以满足高层的要求；而当网络层服务质量较好时，它只须进行很少的工作。另外，它还要处理端到端的差错控制和流量控制等问题，最终为会话提供可靠的、无误的数据传输。

传输层协议的主要内容包括：TCP、UDP、SPX 等。

在 IP 协议栈中第四层是 TCP（传输控制协议）和 UDP（用户数据报协议）所在的协议层。TCP 和 UDP 包含端口号，它可以唯一区分每个数据包包含哪些应用协议（例如 HTTP、FTP、telnet 等）。TCP/UDP 端口号提供的附加信息可以为网络交换机所利用，四层交换机利用这种信息来区分包中的数据，这是第四层交换的基础。

OSI 模型的第五层：会话层

会话层负责在网络中的两节点之间建立和维持通信，并保持会话同步，它还决定通信是否中断，以及通信中断时决定从何处重新发送。

OSI 模型的第六层：表示层

表示层的作用是管理数据的解密与加密，如常见的系统口令处理，当你的账户数据在发送前被加密，在网络的另一端，表示层将对接收到的数据解密。另外，表示层还要对图片和文件格式信息进行解码和编码。

OSI 模型的第七层：应用层

简单来说，应用层就是为操作系统或网络应用程序提供访问网络服务的接口，包括文件传输、文件管理及电子邮件等的信息处理。

应用层协议的代表包括：Telnet、FTP、HTTP、SNMP 等。

www.TopSage.com

OSI 网络模型如图 6-3 所示。

现代负载均衡技术通常操作于 OSI 网络模型的第四层或第七层。第四层负载均衡将一个 Internet 上合法注册的 IP 地址映射为多个内部服务器的 IP 地址，对每次 TCP 连接请求动态使用其中一个内部 IP 地址，达到负载均衡的目的。在第四层交换机中，此种均衡技术得到广泛的应用，一个目标地址是服务器群 VIP（虚拟 IP，Virtual IP address）连接请求的数据包流经交换机，交换机根据源端和目的 IP 地址、TCP 或 UDP 端口号和一定的负载均衡策略，在服务器 IP 和 VIP 间进行映射，选取服务器群中最好的服务器来处理连接请求。

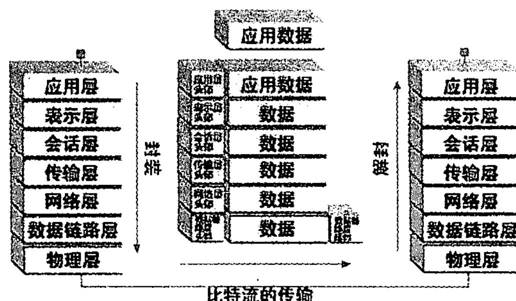


图 6-3 OSI 网络模型

第七层负载均衡控制应用层服务的内容，提供了一种对访问流量的高层控制方式，适合对 HTTP 服务器群的应用。第七层负载均衡技术通过检查流经的 HTTP 报头，根据报头内的信息来执行负载均衡任务。

常见的四/七层负载均衡设备：

1. 硬件四/七层负载均衡交换机

硬件四/七层负载均衡交换机的代表有：F5 BIG-IP、Citrix NetScaler、Radware、Cisco CSS、Foundry 等产品，这些产品价格不菲，高达几十万元人民币。在中国大陆，采用 F5 Network 公司的 BIG-IP 负载均衡交换机的网站（有些网站为部分频道采用）最多，包括：新浪网、雅虎、百度、搜狐、凤凰网、央视国际、中华英才网、猫扑、慧聪网等。

图 6-4 是一张 F5 BIG-IP 实现动、静态网页分离的负载均衡架构图，很好地描述了 F5 BIG-IP 是如何实现四/七层负载均衡的。

（1）如图，假设域名 blog.s135.com 被解析到 F5 的外网/公网虚拟 IP：61.1.1.3（vs_squid），该虚拟 IP 下有一个服务器池（pool_squid），该服务器池下包含两台真实的 Squid 服务器（192.168.1.11 和 192.168.1.12）。

www.TopSage.com

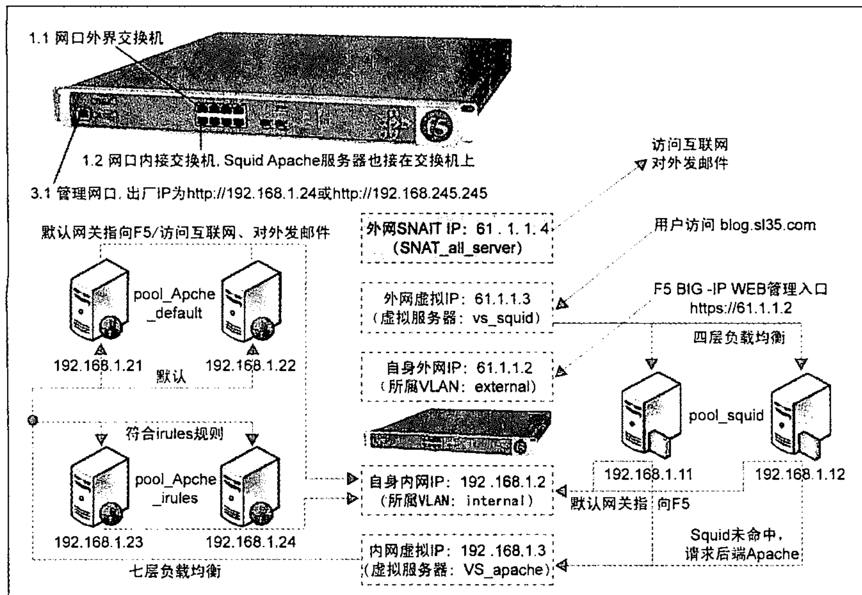


图 6-4　F5 BIG-IP 实现动、静态网页分离的负载均衡架构图

（2）如果 Squid 缓存未命中，则会请求 F5 的内网虚拟 IP：192.168.1.3（vs_apache），该虚拟 IP 下有一个默认服务器池（pool_apache_default），该服务器池下包含两台真实的 Apache 服务器（192.168.1.21 和 192.168.1.22），当该虚拟 IP 匹配 iRules 规则时，则会访问另外一个服务器池（pool_apache_irules），该服务器池下同样包含两台真实的 Apache 服务器（192.168.1.23 和 192.168.1.24）。

（3）另外，所有真实服务器的默认网关指向 F5 的自身内网 IP，即 192.168.1.2。

（4）所有的真实服务器通过 SNAT IP 地址 61.1.1.4 访问互联网。

2. 软件四层负载均衡

软件四层负载均衡的代表作品为 LVS（Linux Virtual Server），作者为曾经在国家并行与分布式处理重点实验室工作，现在已加盟淘宝网的章文嵩博士。LVS 是一个开源的软件，可以实现 LINUX 平台下的简单负载均衡。LVS 集群采用 IP 负载均衡技术和基于内容请求分发技术。调度器具有很好的吞吐率，将请求均衡地转移到不同的服务器上执行，且调度器自动屏蔽掉服务器的故障，从而将一组服务器构成一个高性能的、高可用的虚拟服务器。整个服务器集群的结构对客户是透明的，而且无须修改客户端和服务器端的程序。为此，在设计时要考虑系统的透明性、可伸缩性、高可用性和易管理性。

LVS 负载均衡的结构如图 6-5 所示。

www.TopSage.com

大家网
TopSage.com

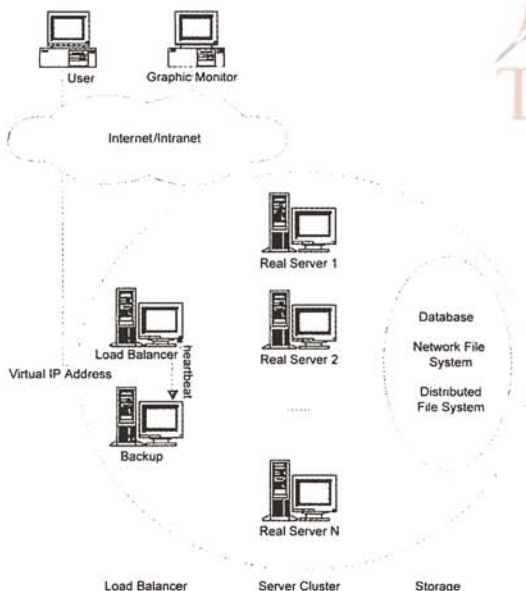


图 6-5 LVS 负载均衡架构图

3. 软件七层负载均衡

软件的七层负载均衡大多基于 HTTP 反向代理方式，代表产品有 Nginx、L7SW（Layer7 switching）、HAProxy 等。Nginx 的反向代理负载均衡能够很好地支持虚拟主机，可配置性很强，可以按轮询、IP 哈希、URL 哈希、权重等多种方式对后端服务器做负载均衡，同时还支持后端服务器的健康检查。

6.2.4 多线多地区智能 DNS 解析与混合负载均衡方式

以新浪首页（www.sina.com.cn）为例，负载均衡同时用到了"多线多地区智能 DNS 解析、DNS 轮询、四/七层负载均衡交换机"等技术。智能 DNS 解析能够根据用户本地设置的 DNS 服务器线路和地区，将对同一个域名请求解析到不同的 IP 上。

例如：当北京电信用户访问 www.sina.com.cn 时，会被新浪的 DNS 服务器解析到北京电信机房的 IP 上；当北京网通用户访问 www.sina.com.cn 时，会被解析到北京网通机房的 IP 上；当教育网的用户访问 www.sina.com.cn 时，会被解析到教育网机房的 IP 上；当广东电信的用户访问 www.sina.com.cn 时，会被解析到广州电信机房的 IP 上；当湖南、湖北的电信用户访问 www.sina.com.cn 时，会被解析到武汉电信机房的 IP 上，等等。

将 DNS 地址设为北京电信的 DNS 服务器 219.141.136.10，通过 Linux 下的 dig 命令可以发现，

www.TopSage.com

访问 www.sina.com.cn 被解析到了北京电信的多台服务器的 IP 上，这属于智能 DNS 解析+DNS 轮询解决负载均衡，如代码 6-2 所示。

代码 6-2

```
[root@localhost ~]# dig www.sina.com.cn

; <<>> DiG 9.3.4-P1 <<>> www.sina.com.cn
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46279
;; flags: qr rd ra; QUERY: 1, ANSWER: 18, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.sina.com.cn.             IN     A

;; ANSWER SECTION:
www.sina.com.cn.       60     IN     CNAME   jupiter.sina.com.cn.
jupiter.sina.com.cn.    600     IN     CNAME   hydra.sina.com.cn.
hydra.sina.com.cn.     20     IN     A      218.30.108.191
hydra.sina.com.cn.     20     IN     A      218.30.108.192
hydra.sina.com.cn.     20     IN     A      218.30.108.63
hydra.sina.com.cn.     20     IN     A      218.30.108.64
hydra.sina.com.cn.     20     IN     A      218.30.108.65
hydra.sina.com.cn.     20     IN     A      218.30.108.66
hydra.sina.com.cn.     20     IN     A      218.30.108.67
hydra.sina.com.cn.     20     IN     A      218.30.108.180
hydra.sina.com.cn.     20     IN     A      218.30.108.181
hydra.sina.com.cn.     20     IN     A      218.30.108.182
hydra.sina.com.cn.     20     IN     A      218.30.108.183
hydra.sina.com.cn.     20     IN     A      218.30.108.184
hydra.sina.com.cn.     20     IN     A      218.30.108.185
hydra.sina.com.cn.     20     IN     A      218.30.108.186
hydra.sina.com.cn.     20     IN     A      218.30.108.187
hydra.sina.com.cn.     20     IN     A      218.30.108.189

;; AUTHORITY SECTION:
sina.com.cn.         72560   IN     NS     ns1.sina.com.cn.
sina.com.cn.         72560   IN     NS     ns2.sina.com.cn.
sina.com.cn.         72560   IN     NS     ns3.sina.com.cn.

;; ADDITIONAL SECTION:
ns1.sina.com.cn.       54070   IN     A      202.106.184.166
ns2.sina.com.cn.       73689   IN     A      61.172.201.254
ns3.sina.com.cn.       57207   IN     A      202.108.44.55

;; Query time: 1 msec
;; SERVER: 219.141.136.10#53(219.141.136.10)
;; WHEN: Sun Aug 16 23:54:43 2009
;; MSG SIZE  rcvd: 433
```


www.TopSage.com

将 DNS 地址设为北京网通的 DNS 服务器 202.106.0.20，通过 Linux 下的 dig 命令可以发现，访问 www.sina.com.cn 最终被解析到了北京网通的一个 IP 地址 202.108.33.32 上，这属于用智能 DNS 解析+四/七层负载均衡交换机解决负载均衡，该 IP 地址 202.108.33.32 是四/七层负载均衡交换机的虚拟 IP，如代码 6-3 所示。

代码 6-3

```
[root@localhost ~]# dig www.sina.com.cn

; <<>> DiG 9.3.4-P1 <<>> www.sina.com.cn
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65261
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.sina.com.cn.            IN     A

;; ANSWER SECTION:
www.sina.com.cn.       24     IN     CNAME   jupiter.sina.com.cn.
jupiter.sina.com.cn.   320    IN     A       202.108.33.32

;; Query time: 2 msec
;; SERVER: 202.106.0.20#53(202.106.0.20)
;; WHEN: Sun Aug 16 23:58:10 2009
;; MSG SIZE  rcvd: 71
```

6.3 Nginx 负载均衡与反向代理的配置实例

6.3.1 完整的 Nginx 反向代理示例

完整的 Nginx 反向代理示例如代码 6-4 所示。

代码 6-4

```
user  www www;

worker_processes 10;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

worker_rlimit_nofile 51200;

events
```

www.TopSage.com




```
{
 use epoll;
 worker_connections 51200;
}

http
{
 include       mime.types;
 default_type  application/octet-stream;

 #charset  utf-8;

 server_names_hash_bucket_size 128;
 client_header_buffer_size 32k;
 large_client_header_buffers 4 32k;

 sendfile on;
 #tcp_nopush     on;

 keepalive_timeout 65;

 tcp_nodelay on;

 fastcgi_connect_timeout 300;
 fastcgi_send_timeout 300;
 fastcgi_read_timeout 300;
 fastcgi_buffer_size 64k;
 fastcgi_buffers 4 64k;
 fastcgi_busy_buffers_size 128k;
 fastcgi_temp_file_write_size 128k;

 gzip on;
 gzip_min_length  1k;
 gzip_buffers     4 16k;
 gzip_http_version 1.1;
 gzip_comp_level 2;
 gzip_types       text/plain application/x-javascript text/css application/xml;
 gzip_vary on;

 #limit_zone  crawler  $binary_remote_addr  10m;

 #允许客户端请求的最大单个文件字节数
 client_max_body_size    300m;

 #缓冲区代理缓冲用户端请求的最大字节数，可以理解为先保存到本地再传给用户
 client_body_buffer_size  128k;

 #跟后端服务器连接的超时时间_发起握手等候响应超时时间
 proxy_connect_timeout    600;

 #连接成功后_等候后端服务器响应时间_其实已经进入后端的排队之中等候处理
 proxy_read_timeout       600;
```


www.TopSage.com

```
#后端服务器数据回传时间_就是在规定时间内后端服务器必须传完所有的数据
proxy_send_timeout      600;

#代理请求缓存区_这个缓存区间会保存用户的头信息以供Nginx进行规则处理_一般只要能保存下头信息即可
proxy_buffer_size       16k;

#同上 告诉Nginx保存单个用的几个Buffer 最大用多大空间
proxy_buffers          4 32k;

#如果系统很忙的时候可以申请更大的proxy_buffers 官方推荐*2
proxy_busy_buffers_size 64k;

#proxy缓存临时文件的大小
proxy_temp_file_write_size 64k;

upstream php_server_pool {
  server   192.168.1.10:80 weight=4 max_fails=2 fail_timeout=30s;
  server   192.168.1.11:80 weight=4 max_fails=2 fail_timeout=30s;
  server   192.168.1.12:80 weight=2 max_fails=2 fail_timeout=30s;
}

upstream message_server_pool {
  server   192.168.1.13:3245;
  server   192.168.1.14:3245 down;
}

upstream bbs_server_pool {
  server   192.168.1.15:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.16:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.17:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.18:80 weight=1 max_fails=2 fail_timeout=30s;
}

#第一个虚拟主机，反向代理php_server_pool这组服务器
server
{
  listen      80;
  server_name  www.yourdomain.com;

  location /
  {
      #如果后端的服务器返回502、504、执行超时等错误，自动将请求转发到upstream负载均衡池中的
       另一台服务器，实现故障转移。
      proxy_next_upstream http_502 http_504 error timeout invalid_header;
      proxy_pass http://php_server_pool;
      proxy_set_header Host  www.yourdomain.com;
      proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log  /data1/logs/www.yourdomain.com_access.log;
}
```

www.TopSage.com

```
#第二个虚拟主机
server
{
  listen      80;
  server_name  www1.yourdomain.com;

  #访问 http://www1.yourdomain.com/message/***地址，反向代理 message_server_pool 这组服
   务器
  location /message/
  {
       proxy_pass http://message_server_pool;
       proxy_set_header Host  $host;
  }

  #访问除了/message/之外的 http://www1.yourdomain.com/***地址，反向代理 php_server_pool
   这组服务器
  location /
  {
       proxy_pass http://php_server_pool;
       proxy_set_header Host  $host;
       proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log  /data1/logs/message.yourdomain.com_access.log;
 }

#第三个虚拟主机
server{
  listen   80;
  server_name  bbs.yourdomain.com *.bbs.yourdomain.com;

  location /
  {
       proxy_pass http://bbs_server_pool;
       proxy_set_header Host  $host;
       proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log off;
 }
}
```

通过上述示例，我们已经看到 Nginx 对于多个域名的负载均衡是如何配置的。Upstream 指令用于设置一组可以在 proxy_pass 和 fastcgi_pass 指令中使用的代理服务器，默认的负载均衡方式为轮询。Upstream 模块中的 Server 指令用于指定后端服务器的名称和参数，服务器的名称可以是一个域名、一个 IP 地址、端口号或 UNIX Socket。

而在 server{...}虚拟主机内，可以通过 proxy_pass 和 fastcgi_pass 指令设置进行反向代理的 upstream 服务器集群。

www.TopSage.com

大家网 TopSage.com

proxy_set_header 指令用于在向反向代理的后端 Web 服务器发起请求时添加指定的 Header 头信息。

当后端 Web 服务器上有多个基于域名的虚拟主机时，要通过添加 Header 头信息 Host，用于指定请求的域名，这样后端 Web 服务器才能识别该反向代理访问请求由哪一个虚拟主机来处理。

使用反向代理之后，后端 Web 服务器（以 PHP 为例）就不能直接通过$_SERVER["REMOTE_ADDR"]变量来获取用户的真实 IP 了，通过$_SERVER["REMOTE_ADDR"]获取的将是 Nginx 负载均衡服务器的 IP。这时，就要通过在 Nginx 反向代理时添加 Header 头信息 X-Forwarded-For，让后端 Web 服务器（以 PHP 为例）能够通过$_SERVER["HTTP_X_FORWARDED_FOR"]获取到用户的真实 IP。

6.3.2 Nginx 负载均衡与反向代理实现动、静态网页分离

动、静态网页分离，就是让动态 PHP 等程序网页去访问 PHP Web 服务器，让缓存页、图片、JavaScript、CSS、Flash 去访问 Squid 等缓存服务器。

在配置 Nginx 负载均衡与反向代理之前，我们首先来看一看 NetScaler 等硬件四/七层负载均衡交换机是如何实现负载均衡的，如图 6-6 所示。

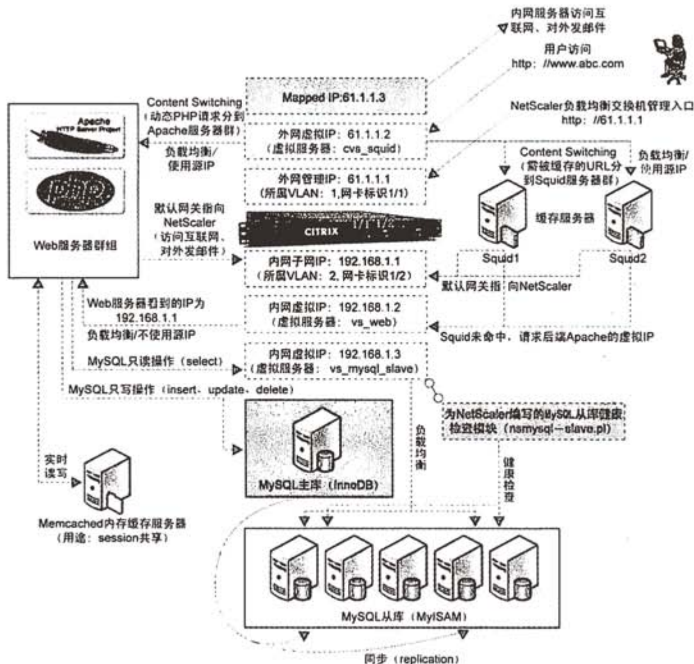


图 6-6 NetScaler 负载均衡交换机动静分离系统架构图

NetScarler 的七层负载均衡是基于 TCP 的，可以用于 Web Server、MySQL 数据库、邮件服务器等大多数基于 TCP 服务器的负载均衡。Nginx 的七层负载均衡仅支持 HTTP、邮件协议，能够实现 Web Server、邮件服务器的负载均衡。在 Web 服务器集群应用中，可以通过 Nginx 负载均衡，来实现并代替 NetScaler 七层负载均衡的部分功能。对于以上的 NetScaler 负载均衡架构，我们可以用以下的 Nginx 负载均衡架构来代替其中的 Web 服务器部分，如图 6-7 所示。

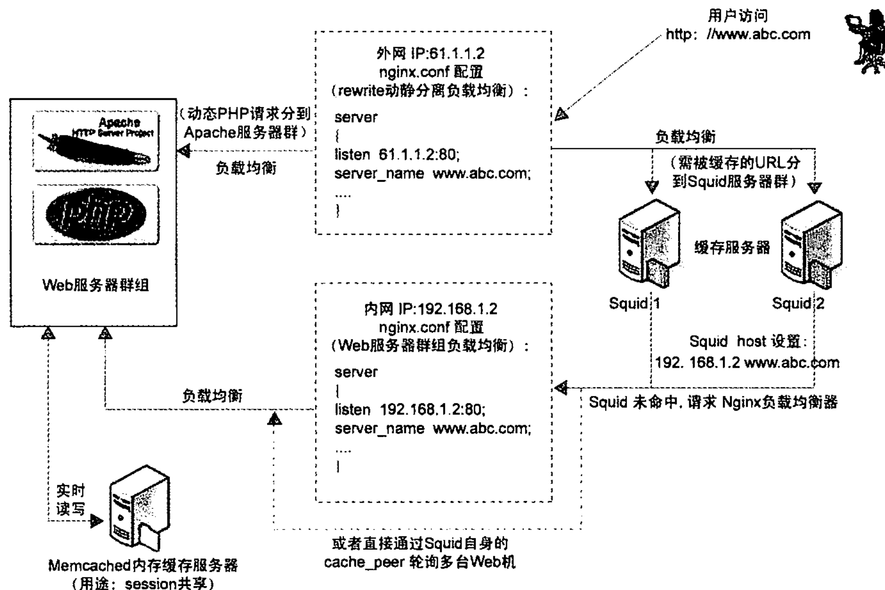


图 6-7 Nginx 反向代理负载均衡动静分离系统架构图

6.4 Nginx 负载均衡的 HTTP Upstream 模块

Upstream 模块是 Nginx 负载均衡的主要模块，它提供了一个简单方法来实现在轮询和客户端 IP 之间的后端服务器负载均衡，并可以对后端服务器进行健康检查。

代码示例如 6-5 所示：

代码 6-5

```
upstream backend {
  server backend1.example.com weight=5;
  server backend2.example.com:8080;
  server unix:/tmp/backend3;
}

server {
```


www.TopSage.com

```
  location / {
    proxy_pass  http://backend;
  }
}
```

6.4.1 ip_hash 指令

ip_hash

语法：ip_hash

默认值：none

使用环境：upstream

当对后端的多台动态应用服务器做负载均衡时，ip_hash 指令能够将某个客户端 IP 的请求通过哈希算法定位到同一台后端服务器上。这样，当来自某个 IP 的用户在后端 Web 服务器 A 上登录后，再访问该站点的其他 URL，能保证其访问的还是后端 Web 服务器 A。如果不采用 ip_hash 指令，假设来自某个 IP 的用户在后端 Web 服务器 A 上登录后，再访问该站点的其他 URL，有可能被定向到后端 Web 服务器 B，C，…上，由于用户登录后 SESSION 信息是记录在服务器 A 上的，B，C，…上没有，这时就会提示用户未登录。

使用 ip_hash 指令无法保证后端服务器的负载均衡，可能有些后端服务器接收到的请求多，有些后端服务器接收到的请求少，而且设置后端服务器权重等方法将不起作用。所以，如果后端的动态应用服务器能够做到 SESSION 共享，还是建议采用后端服务器的 SESSION 共享方式来代替 Nginx 的 ip_hash 方式。

如果后端服务器有时要从 Nginx 负载均衡（已使用 ip_hash）中摘除一段时间，你必须将其标记为“down”，而不是直接从配置文件中删掉或注释掉该后端服务器的信息。代码示例如 6-6：

代码 6-6

```
upstream backend {
  ip_hash;
  server   backend1.example.com;
  server   backend2.example.com;
  server   backend3.example.com  down;
  server   backend4.example.com;
}
```

这样，当原来为 4 台后端服务器时，摘除 backend3.example.com（标记为“down”）后，Nginx 仍然会按 4 台服务器进行哈希。如果直接注释掉“server backend3.example.com”这行，Nginx 就会按照 3 台服务器进行重新哈希，原来被哈希到 backend1.example.com 的客户端 IP 有可能被哈希到 backend2.example.com 服务器上，原有的 SESSION 就会失效。

www.TopSage.com

大家网 TopSage.com

6.4.2 server 指令

server

语法：server name [parameters]

默认值：none

使用环境：upstream

该指令用于指定后端服务器的名称和参数。服务器的名称可以是一个域名、一个 IP 地址、端口号或 UNIX Socket。

在后端服务器名称之后，可以跟以下参数：

weight = NUMBER——设置服务器的权重，权重数值越高，被分配到的客户端请求数越多。如果没有设置权重，则为默认权重 1。

max_fails = NUMBER——在参数 fail_timeout 指定的时间内对后端服务器请求失败的次数，如果检测到后端服务器无法连接及发生服务器错误（404 错误除外），则标记为失败。如果没有设置，则为默认值 1。设为数值 0 将关闭这项检查。

fail_timeout = TIME——在经历参数 max_fails 设置的失败次数后，暂停的时间。

down——标记服务器为永久离线状态，用于 ip_hash 指令。

backup——仅仅在非 backup 服务器全部宕机或繁忙的时候才启用。

示例如下：

```
upstream backend {
 server  backend1.example.com   weight=5;
 server  127.0.0.1:8080        max_fails=3  fail_timeout=30s;
 server  unix:/tmp/backend3;
}
```

6.4.3 upstream 指令

upstream

语法：upstream name { ... }

默认值：none

使用环境：http

该指令用于设置一组可以在 proxy_pass 和 fastcgi_pass 指令中使用的代理服务器，默认的负

www.TopSage.com

载均衡方式为轮询。示例如下：

```
upstream backend {
  server backend1.example.com weight=5;
  server 127.0.0.1:8080      max_fails=3  fail_timeout=30s;
  server unix:/tmp/backend3;
}
```

6.4.4 upstream 相关变量

从 Nginx 0.5.18 版本开始，可以支持用 log_format 指令设置日志格式，日志格式中可以使用变量，例如：

```
log_format timing '$remote_addr - $remote_user [$time_local]  $request '
  'upstream_response_time $upstream_response_time '
  'msec $msec request_time $request_time';

log_format up_head '$remote_addr - $remote_user [$time_local]  $request '
  'upstream_http_content_type $upstream_http_content_type';
```

upstream 模块拥有以下变量：

$upstream_addr

处理请求的 upstream 服务器地址。

$upstream_status

Upstream 服务器的应答状态。

$upstream_response_time

Upstream 服务器响应时间（毫秒），多个响应以逗号和冒号分割。

$upstream_http_$HEADER

任意的 HTTP 协议头信息，例如：

```
$upstream_http_host
```

6.5 Nginx 负载均衡服务器的双机高可用

如果将 Web 服务器集群当作一个城池，那么负载均衡服务器则相当于城门，重要性不言而喻。如果“城门”关闭了，与外界的通道也就掐断了。如果只有一台 Nginx 负载均衡服务器，当该服务器发生故障时，则会导致整个网站无法访问。因此，我们需要两台以上的 Nginx 负载均衡服务器，实现故障转移与高可用。

双机高可用一般是通过虚拟 IP（也称漂移 IP）方式来实现的，基于 Linux/Unix 的 IP 别名技术。双机高可用方式目前可分为两种：第一种方式为一台主服务器加一台热备服务器，正常情况下主服务器绑定一个公网虚拟 IP，提供负载均衡服务，热备服务器处于空闲状态，当主服务器发生故障时，热备服务器接管主服务器的虚拟 IP，提供负载均衡服务；第二种方式为两台负载均衡服务器都处于活动状态，各自绑定一个公网虚拟 IP，提供负载均衡服务，当其中一台服务器发生故障时，另一台服务器接管发生故障服务器的虚拟 IP。第一种方式较为常见，但始终有一台服务器处于空闲状态，浪费了一台服务器的负载均衡处理能力。第二种方式需要多用一个公网 IP，笔者已经在金山游戏官方网站——逍遥网（xoyo.com）线上环境成功使用，能够在正常情况下将两台服务器都用于实际的负载均衡处理。

第一种方式：

（1）www.yourdomain.com 域名解析到虚拟 IP 61.1.1.2 上。

（2）正常情况下，主机 61.1.1.4 绑定虚拟 IP 61.1.1.2。

```
/sbin/ifconfig eth0:1 61.1.1.2 broadcast  61.1.1.255 netmask 255.255.255.0 up
/sbin/route add -host 61.1.1.2 dev eth0:1
/sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1
```

（3）用户访问 www.yourdomain.com（虚拟 IP 61.1.1.2）实际访问的是主机 61.1.1.4，而备机 61.1.1.5 则处于空闲状态。

（4）如果主机 61.1.1.4 发生故障，备机 61.1.1.5 将在几秒钟内接管虚拟 IP 61.1.1.2，与自己绑定，并发送 ARPing 包给 IDC 的公网网关刷新 MAC 地址。

```
/sbin/ifconfig eth0:1 61.1.1.2 broadcast  61.1.1.255 netmask 255.255.255.0 up
/sbin/route add -host 61.1.1.2 dev eth0:1
/sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1
```

（5）这时，用户访问 www.yourdomain.com（虚拟 IP 61.1.1.2）实际上访问的是备机 61.1.1.5，从而实现故障转移与高可用，避免了单点故障。转移过程如图 6-8 所示。

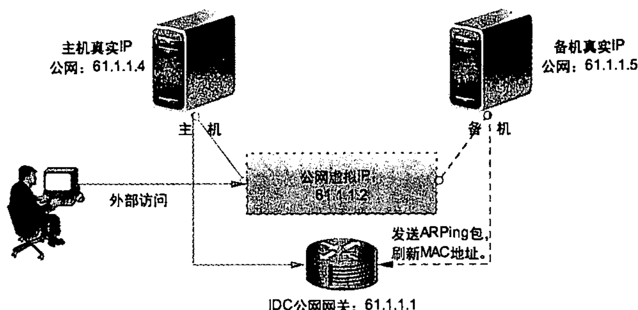


图 6-8　一台主机配一台备机的可用服务方式

www.TopSage.com

大家网 TopSage.com

另外，第一种方式还可以利用基于 VRRP 路由协议的 Keepalived 软件来实现。

第二种方式：

（1）www.yourdomain.com 域名通过 DNS 轮询解析到虚拟 IP 61.1.1.2 和 61.1.1.3 上。

（2）正常情况下，服务器①61.1.1.4 绑定虚拟 IP 61.1.1.2，服务器②61.1.1.5 绑定虚拟 IP 61.1.1.3。其联结与运行方式如图 6-9 所示。

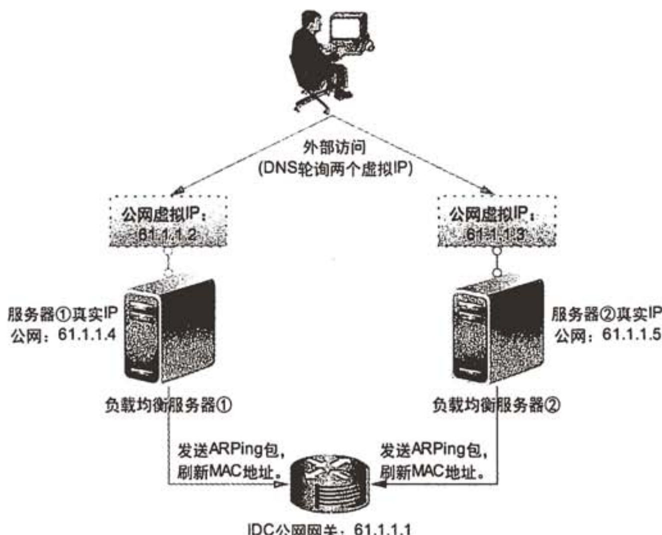


图 6-9　两台负载均衡服务器的高可用服务方式（正常状态）

在服务器①61.1.1.4 上执行以下命令：

```
/sbin/ifconfig eth0:1 61.1.1.2 broadcast  61.1.1.255 netmask 255.255.255.0 up
/sbin/route add -host 61.1.1.2 dev eth0:1
/sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1
```

在服务器②61.1.1.5 上执行以下命令：

```
/sbin/ifconfig eth0:1 61.1.1.3 broadcast  61.1.1.255 netmask 255.255.255.0 up
/sbin/route add -host 61.1.1.3 dev eth0:1
/sbin/arping -I eth0 -c 3 -s 61.1.1.3 61.1.1.1
```

（3）用户访问 www.yourdomain.com（虚拟 IP 61.1.1.2 和 61.1.1.3）实际上是根据 DNS 轮询访问两台负载均衡服务器，两台服务器均处于活动状态。

（4）如果服务器①发生故障，服务器②将在几秒钟内接管服务器①的虚拟 IP 61.1.1.2，与自己绑定，并发送 ARPing 包给 IDC 的公网网关刷新 MAC 地址。这时，服务器②同时绑定 61.1.1.2 和 61.1.1.3 两个虚拟 IP，其联结与运行方式如图 6-10 所示。

www.TopSage.com

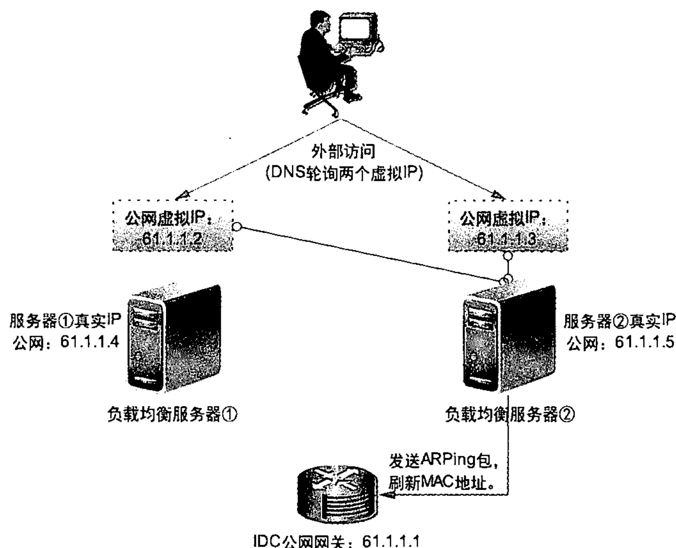


图 6-10 其中一台负载均衡服务器发生故障的运行方式

在服务器②61.1.1.5 上执行以下命令：

```
/sbin/ifconfig eth0:1 61.1.1.2 broadcast  61.1.1.255 netmask 255.255.255.0 up
/sbin/route add -host 61.1.1.2 dev eth0:1
/sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1
```

我们可以写两个 shell 脚本，来实现第二种方式的自动故障转移。

以下代码 6-7 为脚本 1（nginx_ha1.sh），部署在 Nginx 负载均衡服务器①：

代码 6-7

```
#!/bin/sh
LANG=C
date=$(date -d "today" +"%Y-%m-%d %H:%M:%S")

function_bind_vip1()
{
   /sbin/ifconfig eth0:ha1 61.1.1.2 broadcast 219.232.254.255 netmask
    255.255.255.192 up
   /sbin/route add -host 61.1.1.2 dev eth0:ha1
}

function_bind_vip2()
{
   /sbin/ifconfig eth0:ha2 61.1.1.3 broadcast 219.232.254.255 netmask
    255.255.255.192 up
   /sbin/route add -host 61.1.1.3 dev eth0:ha2
}
```


www.TopSage.com

```
function_restart_nginx()
{
   kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
}

function_remove_vip1()
{
   /sbin/ifconfig eth0:ha1 61.1.1.2 broadcast 219.232.254.255 netmask
    255.255.255.192 down
}

function_remove_vip2()
{
   /sbin/ifconfig eth0:ha2 61.1.1.3 broadcast 219.232.254.255 netmask
    255.255.255.192 down
}

function_vip_arping1()
{
   /sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1 > /dev/null 2>&1
}

function_vip_arping2()
{
   /sbin/arping -I eth0 -c 3 -s 61.1.1.3 61.1.1.1 > /dev/null 2>&1
}

bind_time_vip1="N";
bind_time_vip2="N";

while true
do
   httpcode_rip1=`/usr/bin/curl -o /dev/null -s -w %{http_code} http://61.1.1.4`
   httpcode_rip2=`/usr/bin/curl -o /dev/null -s -w %{http_code} http://61.1.1.5`

   if [ x$httpcode_rip1 == "x200" ];
   then
      if [ $bind_time_vip1 == "N" ];
      then
         function_bind_vip1
         function_vip_arping1
         function_restart_nginx
         bind_time_vip1="Y"
      fi
      function_vip_arping1
   else
      if [ $bind_time_vip1 == "Y" ];
      then
         function_remove_vip1
         bind_time_vip1="N"
      fi
   fi
```

www.TopSage.com

大家网
TopSage.com

```
   if [ x$httpcode_rip2 == "x200" ];
   then
      if [ $bind_time_vip2 == "Y" ];
      then
         function_remove_vip2
         bind_time_vip2="N"
      fi
   else
      if [ $bind_time_vip2 == "N" ];
      then
         function_bind_vip2
         function_vip_arping2
         function_restart_nginx
         bind_time_vip2="Y"
      fi
      function_vip_arping2
   fi

   sleep 5
done
```

在 Nginx 负载均衡服务器①将脚本驻留后台运行：

```
nohup /bin/sh ./nginx_ha1.sh 2>&1 > /dev/null &
```

以下代码 6-8 为脚本 2（server2.sh），部署在 Nginx 负载均衡服务器②：

代码 6-8

```
#!/bin/sh
LANG=C
date=$(date -d "today" +"%Y-%m-%d %H:%M:%S")

function_bind_vip1()
{
   /sbin/ifconfig eth0:ha1 61.1.1.3 broadcast 219.232.254.255 netmask
    255.255.255.192 up
   /sbin/route add -host 61.1.1.3 dev eth0:ha1
}

function_bind_vip2()
{
   /sbin/ifconfig eth0:ha2 61.1.1.2 broadcast 219.232.254.255 netmask
    255.255.255.192 up
   /sbin/route add -host 61.1.1.2 dev eth0:ha2
}

function_restart_nginx()
{
   kill -USR1 `cat /usr/local/webserver/nginx/nginx.pid`
}
```


www.TopSage.com

```
function_remove_vip1()
{
   /sbin/ifconfig eth0:ha1 61.1.1.3 broadcast 219.232.254.255 netmask
    255.255.255.192 down
}

function_remove_vip2()
{
   /sbin/ifconfig eth0:ha2 61.1.1.2 broadcast 219.232.254.255 netmask
    255.255.255.192 down
}

function_vip_arping1()
{
   /sbin/arping -I eth0 -c 3 -s 61.1.1.3 61.1.1.1 > /dev/null 2>&1
}

function_vip_arping2()
{
   /sbin/arping -I eth0 -c 3 -s 61.1.1.2 61.1.1.1 > /dev/null 2>&1
}

bind_time_vip1="N";
bind_time_vip2="N";

while true
do
   httpcode_rip1=`/usr/bin/curl -o /dev/null -s -w %{http_code} http://61.1.1.5`
   httpcode_rip2=`/usr/bin/curl -o /dev/null -s -w %{http_code} http://61.1.1.4`

   if [ x$httpcode_rip1 == "x200" ];
   then
      if [ $bind_time_vip1 == "N" ];
      then
         function_bind_vip1
         function_vip_arping1
         function_restart_nginx
         bind_time_vip1="Y"
      fi
      function_vip_arping1
   else
      if [ $bind_time_vip1 == "Y" ];
      then
         function_remove_vip1
         bind_time_vip1="N"
      fi
   fi

   if [ x$httpcode_rip2 == "x200" ];
   then
      if [ $bind_time_vip2 == "Y" ];
      then
```

```
            function_remove_vip2
            bind_time_vip2="N"
        fi
    else
        if [ $bind_time_vip2 == "N" ];
        then
            function_bind_vip2
            function_vip_arping2
            function_restart_nginx
            bind_time_vip2="Y"
        fi
        function_vip_arping2
    fi

    sleep 5
done
```

在 Nginx 负载均衡服务器②将脚本驻留后台运行：

```
nohup /bin/sh ./nginx_ha2.sh 2>&1 > /dev/null &
```


www.TopSage.com

大家网
TopSage.com

第7章 Nginx 的 Rewrite 规则与实例

7.1 什么是 Nginx 的 Rewrite 规则

Rewrite 主要的功能就是实现 URL 的重写，Nginx 的 Rewrite 规则采用 PCRE（Perl Compatible Regular Expressions）Perl 兼容正则表达式的语法进行规则匹配，如果您需要 Nginx 的 Rewrite 功能，在编译 Nginx 之前，须要编译安装 PCRE 库。

正则表达式（英文：Regular Expression）在计算机科学中，是指一个用来描述或匹配一系列符合某个句法规则的字符串的单个字符串。最初的正则表达式出现于理论计算机科学的自动控制理论和形式化语言理论中。在这些领域中有对计算（自动控制）的模型和对形式化语言描述与分类的研究。20 世纪 40 年代，Warren McCulloch 与 Walter Pitts 将神经系统中的神经元描述成小而简单的自动控制元。20 世纪 50 年代，数学家斯蒂芬・科尔・克莱尼利用称之为正则集合的数学符号来描述此模型。肯・汤普逊将此符号系统引入编辑器 QED，然后是 Unix 上的编辑器 ed，并最终引入 grep。自此，正则表达式被广泛使用于各种 Unix 或类似 Unix 的工具，例如 Perl。

通过 Rewrite 规则，可以实现规范的 URL、根据变量来做 URL 转向及选择配置。例如，一些使用 MVC 框架的程序只有一个入口，可以通过 Rewrite 来实现。一些动态 URL 地址须要伪装成静态 HTML，便于搜索引擎抓取，也需要 Rewrite 来处理。一些由于目录结构、域名变化的旧 URL，须要跳转到新的 URL 上，也可以通过 Rewrite 来处理。

www.TopSage.com

7.2 Nginx Rewrite 规则相关指令

Nginx Rewrite 规则相关指令有 if、rewrite、set、return、break 等，其中 rewrite 是最关键的指令。一个简单的 Nginx Rewrite 规则语法如下：

```
rewrite ^/b/(.*)\.html /play.php?video=$1 break;
```

如果加上 if 语句，示例如下：

```
if (!-f $request_filename)
{
rewrite ^/img/(.*)$ /site/$host/images/$1 last;
}
```

7.2.1 break 指令

语法：break

默认值：none

使用环境：server, location, if

该指令的作用是完成当前的规则集，不再处理 rewrite 指令。

示例如下：

```
if ($slow) {
  limit_rate 10k;
  break;
}
```

7.2.2 if 指令

语法：if (condition) { ... }

默认值：none

使用环境：server, location

该指令用于检查一个条件是否符合，如果条件符合，则执行大括号内的语句。if 指令不支持嵌套，不支持多个条件&&和||处理。

以下信息可以被指定为条件：

（1）变量名，错误的值包括：空字符串""，或者任何以 0 开始的字符串；

w.TopSage.com

（2）变量比较可以使用“=”（表示等于）和“!=”（表示不等于）运算符；

（3）正则表达式模式匹配可以使用“~*”和“~”符号；

（4）“~”符号表示区分大小写字母的匹配；

（5）“~*”符号表示不区分大小写字母的匹配（例如 firefox 与 FireFox 是匹配的）；

（6）“!~”和“!~*”符号的作用刚好和“~”、“!~*”相反，表示不匹配；

（7）“-f”和“!-f”用来判断文件是否存在；

（8）“-d”和“!-d”用来判断目录是否存在；

（9）“-e”和“!-e”用来判断文件或目录是否存在；

（10）“-x”和“!-x”用来判断文件是否可执行。

部分正则表达式可以在圆括号“()”内，其值可以通过后面的变量$1 至$9 访问，示例如代码 7-1 所示：

代码 7-1

```
if ($http_user_agent ~ MSIE) {
  rewrite  ^(.*)$  /msie/$1  break;
}

if ($http_cookie ~* "id=([^;] +)(?:;|$)" ) {
  set  $id  $1;
}

if ($request_method = POST ) {
  return 405;
}

if (!-f $request_filename) {
  break;
  proxy_pass  http://127.0.0.1;
}

if ($slow) {
  limit_rate  10k;
}

if ($invalid_referer) {
  return   403;
}

if ($args ^~ post=140){
  rewrite ^ http://example.com/permanent;
}
```



示例代码 7-1 中，内置的变量$ invalid_referer 值由 valid_referers 指令提供。

7.2.3 return 指令

语法：return code

默认值：none

使用环境：server, location, if

该指令用于结束规则的执行并返回状态码给客户端。状态码可以使用这些值：204, 400, 402~406, 408, 410, 411, 413, 416 及 500~504。此外，非标准状态码 444 将以不发送任何 Header 头的方式结束连接。

示例，如果访问的 URL 以“.Sh”和“.Bash”结尾，则返回状态码 403：

```
location ~ .*\.(sh|bash)?$
{
   return 403;
}
```

下面，我们来详细介绍 return 指令支持的状态码。

204 No Content

服务器成功处理了请求，但无须返回任何实体内容，并且希望返回更新了的元信息。响应可能通过实体头部的形式，返回新的或更新后的元信息。如果存在这些头部信息，则应当与所请求的变量相呼应。

如果客户端是浏览器，那么用户浏览器应保留发送了该请求的页面，而不产生任何文档视图上的变化，即使按照规范新的或更新后的元信息，也应当被应用到用户浏览器活动视图中的文档。

400 Bad Request

由于包含语法错误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求。

402 Payment Required

该状态码是为了将来可能的需求而预留的。

403 Forbidden

服务器已经理解请求，但是拒绝执行它。与 401 响应不同的是，身份验证并不能提供任何帮助，而且这个请求也不应该被重复提交。如果这不是一个 HEAD 请求，而且服务器希望能够讲清楚为何请求不能被执行，就应该在实体内描述拒绝的原因。当然服务器也可以返回一个 404 响

www.TopSage.com

应，假如它不希望让客户端获得任何信息。

404 Not Found

请求失败，请求所希望得到的资源未在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况，应当使用 410 状态码来告知旧资源因为某些内部的配置机制问题，已经永久地不可用，而且没有任何可以跳转的地址。404 这个状态码被广泛应用于当服务器不想揭示为何请求被拒绝，或者没有其他适合的响应可用的情况下。

405 Method Not Allowed

请求行中指定的请求方法不能被用于请求相应的资源。该响应必须返回一个 Allow 头信息，用以表示出当前资源能够接受的请求方法的列表。

鉴于 PUT，DELETE 方法会对服务器上的资源进行写操作，因而绝大部分的网页服务器都不支持或在默认配置下不支持上述请求方法，对于此类请求均会返回 405 错误。

406 Not Acceptable

请求的资源的内容特性无法满足请求头中的条件，因而无法生成响应实体。

除非这是一个 HEAD 请求，否则该响应就应当返回一个包含可以让用户或浏览器从中选择最合适的实体特性及地址列表的实体。实体的格式由 Content-Type 头中定义的媒体类型决定。浏览器可以根据格式及自身能力自行作出最佳选择。但是，规范中并没有定义任何作出此类自动选择的标准。

408 Request Timeout

请求超时。客户端没有在服务器预备等待的时间内完成一个请求的发送。客户端可以随时再次提交这一请求而无须进行任何更改。

410 Gone

被请求的资源在服务器上已经不再可用，而且没有任何已知的转发地址。这样的状况应当被认为是永久性的。如果可能，拥有链接编辑功能的客户端应当在获得用户许可后删除所有指向这个地址的引用。如果服务器不知道或无法确定这个状况是否是永久的，就应该使用 404 状态码。除非额外说明，否则这个响应是可缓存的。

410 响应的目的主要是帮助网站管理员维护网站，通知用户该资源已经不再可用，并且服务器拥有者希望所有指向这个资源的远端连接也被删除。这类事件在限时、增值服务中很普遍。同样，410 响应也被用于通知客户端在当前服务器站点上，原本属于某个个人的资源已经不再可用。当然，是否要把所有永久不可用的资源标记为'410 Gone'，以及是否要保持此标记多长时间，完全取决于服务器拥有者。

www.TopSage.com

411 Length Required

服务器拒绝在没有定义 Content-Length 头的情况下接受请求。在添加了表明请求消息体长度的有效 Content-Length 头之后，客户端可以再次提交该请求。

413 Request Entity Too Large

服务器拒绝处理当前请求，因为该请求提交的实体数据大小超过了服务器愿意或能够处理的范围。此种情况下，服务器可以关闭连接以免客户端继续发送此请求。

如果这个状况是临时的，服务器应当返回一个 Retry-After 的响应头，以告知客户端可以在多少时间以后重新尝试。

416 Requested Range Not Satisfiable

如果请求中包含了 Range 请求头，并且 Range 中指定的任何数据范围都与当前资源的可用范围不重合，同时请求中又没有定义 If-Range 请求头，那么服务器就应当返回 416 状态码。

假如 Range 使用的是字节范围，那么这种情况就是指请求指定的所有数据范围的首字节位置都超过了当前资源的长度。服务器也应当在返回 416 状态码的同时，包含一个 Content-Range 实体头，用以指明当前资源的长度。这个响应也被禁止使用 multipart/byteranges 作为其 Content-Type。

500 Internal Server Error

服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。

501 Not Implemented

服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求时。

502 Bad Gateway

作为网关或代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。

503 Service Unavailable

由于临时的服务器维护或过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可以包含一个 Retry-After 头用以标明这个延迟时间。如果没有给出这个 Retry-After 信息，那么客户端应当以处理 500 响应的方式处理它。

注意：503 状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。

www.TopSage.com

大家网 TopSage.com

504 Gateway Timeout

作为网关或代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI 标识出的服务器，例如 HTTP、FTP、LDAP）或辅助服务器（例如 DNS）收到响应。

注意：某些代理服务器在 DNS 查询超时时会返回 400 或 500 错误。

7.2.4 rewrite 指令

语法：rewrite regex replacement flag

默认值：none

使用环境：server, location, if

该指令根据表达式来重定向 URI，或者修改字符串。指令根据配置文件中的顺序来执行。

注意重写表达式只对相对路径有效。如果你想配对主机名，你应该使用 if 语句，代码如下：

```
if ($host ~* www\.(.*)) {
 set $host_without_www $1;
 rewrite ^(.*)$ http://$host_without_www$1 permanent; # $1 contains '/foo', not
   'www.mydomain.com/foo'
}
```

如果替换串以 *http://*开头，将会采用 301 或 302 跳转进行 URL 重定向。

rewrite 指令的最后一项参数为 flag 标记，支持的 flag 标记有：

- last——相当于 Apache 里的[L]标记，表示完成 rewrite；
- break——本条规则匹配完成后，终止匹配，不再匹配后面的规则；
- redirect——返回 302 临时重定向，浏览器地址栏会显示跳转后的 URL 地址；
- permanent——返回 301 永久重定向，浏览器地址栏会显示跳转后的 URL 地址。

在以上的标记中，last 和 break 用来实现 URI 重写，浏览器地址栏的 URL 地址不变，但在服务器端访问的路径发生了变化。redirect 和 permanent 用来实现 URL 跳转，浏览器地址栏会显示跳转后的 URL 地址。

last 和 break 标记的实现功能类似，但二者之间有细微的差别，使用 alias 指令时必须用 last 标记，使用 proxy_pass 指令时要使用 break 标记。last 标记在本条 rewrite 规则执行完毕后，会对其所在的 server{......}标签重新发起请求，而 break 标记则在本条规则匹配完成后，终止匹配，不再匹配后面的规则。例如以下这段规则，就必须使用 break 标记，使用 last 标记会导致死循环：

www.TopSage.com

```
location /cms/ {
  proxy_pass http://test.yourdomain.com;
  rewrite "^/cms/(.*)\.html$" /cms/index.html break;
}
```

因此，一般在根 location 中（即 location /{......}）或直接在 server 标签中编写 rewrite 规则，推荐使用 last 标记，在非根 location 中（例如 location /cms/{......}），则使用 break 标记。例如：

```
rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 last;
rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra  last;
return  403;

location /download/ {
  rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
  rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra  break;
  return  403;
}
```

如果被替换的 URI 中含有参数（即类似/app/test.php?id=5 之类的 URI），默认情况下参数会被自动附加到替换串上，你可以通过在替换串的末尾加上?标记来解决这一问题。

```
rewrite ^/users/(.*)$ /show?user=$1? last;
```

下面我们来比较一下，不加?标记和加上?标记的 URL 跳转区别：

```
rewrite ^/test(.*)$ http://www.yourdomain.com/home permanent;
```

访问 *http://www.yourdomain.com/test?id=5* 经过 301 跳转后的 URL 地址为 *http://www.yourdomain.com/home?id=5*

```
rewrite ^/test(.*)$ http://www.yourdomain.com/home? permanent;
```

访问 *http://www.yourdomain.com/test?id=5* 经过 301 跳转后的 URL 地址为 *http://www.yourdomain.com/home*

注：对花括号({ 和 })来说，它们既能用在重定向的正则表达式里，也能用在配置文件里分割代码块，为了避免冲突，正则表达式里如果带花括号，应该用双引号（或者单引号）包围。比如，要将类似以下的 URL：

/photos/123456

重定向到：

/path/to/photos/12/1234/123456.png

可以用以下方法（注意双引号）：

```
rewrite "/photos/([0-9] {2})([0-9] {2})([0-9] {2})"
/path/to/photos/$1/$1$2/$1$2$3.png;
```


www.TopSage.com

7.2.5 set 指令

语法：set variable value

默认值：none

使用环境：server, location, if

该指令用于定义一个变量，并给变量赋值。变量的值可以为文本、变量及文本变量的联合。

示例如下：

```
set $varname 'hello';
```

7.2.6 uninitialized_variable_warn 指令

语法：uninitialized_variable_warn on|off

默认值：uninitialized_variable_warn on

使用环境：http, server, location, if

该指令用于开启或关闭记录关于未初始化变量的警告信息，默认值为开启。

7.2.7 Nginx Rewrite 可以用到的全局变量

在 if、location、rewrite 指令中，可以使用以下全局变量：

- $args
- $content_length
- $content_type
- $document_root
- $document_uri
- $host
- $http_user_agent
- $http_cookie
- $limit_rate
- $request_body_file
- $request_method
- $remote_addr

大家网 TopSage.com

- $remote_port
- $remote_user
- $request_filename
- $request_uri
- $query_string
- $scheme
- $server_protocol
- $server_addr
- $server_name
- $server_port
- $uri

7.3 PCRE 正则表达式语法

Perl 正则表达式源自于 Henry Spencer 写的 regex，它已经演化成了 PCRE（Perl 兼容正则表达式，Perl Compatible Regular Expressions），一个由 Philip Hazel 开发的，为很多现代工具所使用的库。在 Nginx 的 rewrite 指令的语法中，“rewrite regex replacement flag”之中的 regex 使用的是 PCRE 正则表达式。表 7-1 是在 PCRE 中元字符及其在正则表达式上下文中行为的一个完整列表。

表 7-1 PCRE 正则表达式语法一览表

字符	描述
\	将下一个字符标记为一个特殊字符，或一个原义字符，或一个向后引用，或一个八进制转义符。例如，“\n”匹配一个换行符。序列“\\”匹配“\”而“\(”则匹配“(”
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配“\n”或“\r”之后的位置
$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$也匹配“\n”或“\r”之前的位置
*	匹配前面的子表达式零次或多次。例如，zo*能匹配“z”及“zoo”。*等价于{0,}
+	匹配前面的子表达式一次或多次。例如，“zo+”能匹配“zo”及“zoo”，但不能匹配“z”。+等价于{1,}
?	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”中的“do”。?等价于{0,1}
?	当该字符紧跟在任何一个其他限制符(*,+,?,{n},{n,},{n,m})后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+?”将匹配单个“o”，而“o+”将匹配所有“o”

www.TopSage.com

续表

字符	描述
{n}	n 是一个非负整数。匹配确定的 n 次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个 o
{n,}	n 是一个非负整数。至少匹配 n 次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有 o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o*”
{n,m}	m 和 n 均为非负整数，其中 n<=m。最少匹配 n 次且最多匹配 m 次。例如，“o{1,3}”将匹配“fooooood”中的前三个 o。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格
.	匹配除“\n”之外的任何单个字符。要匹配包括“\n”在内的任何字符，请使用像“[.\n]”的模式
(pattern)	匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 SubMatches 集合，在 JScript 中则使用$0…$9 属性。要匹配圆括号字符，请使用“\(”或“\)”
(?:pattern)	匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用“或”字符(|)来组合一个模式的各个部分是很有用的。例如: 'industr(?:y|ies)'就是一个比'industry|industries'更简略的表达式
(?=pattern)	正向预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配无须获取供以后使用。例如，“Windows(?=95|98|NT|2000)”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在本次匹配搜索的最后一次匹配，而不是从包含预查的字符之后开始
(?!pattern)	负向预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配无须获取供以后使用。例如“Windows(?!95|98|NT|2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
x|y	匹配 x 或 y。例如，“z|food”能匹配“z”或“food”。“(z|f)ood”则匹配“zood”或“food”
[xyz]	字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”
[^xyz]	负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“p”
[a-z]	字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符
[^a-z]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”

www.TopSage.com

续表

字符	描述
\B	匹配非单词边界。"er\B"能匹配"verb"中的"er"，但不能匹配"never"中的"er"
\cx	匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的"c"字符
\d	匹配一个数字字符。等价于[0-9]
\D	匹配一个非数字字符。等价于[^0-9]
\f	匹配一个换页符。等价于\x0c 和\cL
\n	匹配一个换行符。等价于\x0a 和\cJ
\r	匹配一个回车符。等价于\x0d 和\cM
\s	匹配任何空白字符，包括空格、制表符、换页符等。等价于[\f\n\r\t\v]
\S	匹配任何非空白字符。等价于[^\f\n\r\t\v]
\t	匹配一个制表符。等价于\x09 和\cI
\v	匹配一个垂直制表符。等价于\x0b 和\cK
\w	匹配包括下划线的任何单词字符。等价于"[A-Za-z0-9_]"
\W	匹配任何非单词字符。等价于"[^A-Za-z0-9_]"
\xn	匹配 n，其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，"\x41"匹配"A"。"\x041"则等价于"\x04"&"1"。正则表达式中可以使用 ASCII 编码
\num	匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。例如，"(.)\1"匹配两个连续的相同字符
\n	标识一个八进制转义值或一个向后引用。如果\n 之前至少 n 个获取的子表达式，则 n 为向后引用。否则，如果 n 为八进制数字(0~7)，则 n 为一个八进制转义值
\nm	标识一个八进制转义值或一个向后引用。如果\nm 之前至少有 nm 个获取的子表达式，则 nm 为向后引用。如果\nm 之前至少有 n 个获取的子表达式，则 n 为一个后跟文字 m 的向后引用。如果前面的条件都不满足，若 n 和 m 均为八进制数字(0~7)，则\nm 将匹配八进制转义值 nm
\nml	如果 n 为八进制数字(0~3)，且 m 和 l 均为八进制数字(0~7)，则匹配八进制转义值 nml
\un	匹配 n，其中 n 是一个用 4 个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号（©）

7.4 Nginx 的 Rewrite 规则编写实例

文件和目录不存在时，重定向到某个 PHP 文件上，适用于 WordPress 等 MVC 结构的开源博客系统：

www.TopSage.com




```
if (!-e $request_filename) {
   rewrite ^/(.*)$  /index.php  last;
}
```

多目录转成参数 abc.domian.com/sort/2 => abc.domian.com/index.php?act=sort&name= abc&id=2：

```
if ($host ~* (.*)\.domain\.com) {
   set $sub_name $1;
   rewrite ^/sort\/(\d+)\/?$ /index.php?act=sort&cid=$sub_name&id=$1 last;
}
```

目录对换/123456/xxxx -> /xxxx?id=123456：

```
rewrite ^/(\d+)/(.+)/ /$2?id=$1 last;
```

如果客户端使用 IE 浏览器，则重定向到/nginx-ie 目录下：

```
if ($http_user_agent ~ MSIE) {
   rewrite ^(.*)$ /nginx-ie/$1 break;
}
```

禁止访问多个目录：

```
location ~ ^/(cron|templates)/ {
   deny all;
   break;
}
```

禁止访问以/data 开头的文件：

```
location ~ ^/data {
   deny all;
}
```

设置某些类型文件的浏览器缓存时间：

```
location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
{
   expires     30d;
}
location ~ .*\.(js|css)?$
{
   expires     1h;
}
```

将多级目录下的文件转换成一个文件/job-123-456-789.html 指向/job/123/456/789.html：

```
rewrite ^/job-([0-9]+)-([0-9]+)-([0-9]+)\.html$ /job/$1/$2/jobshow_$3.html last;
```

禁止访问以.sh、.flv、.mp4 为文件名后缀的 URL 地址：

```
location ~ .*\.(sh|flv|mp4)?$ { return 403; }
```

#适用于 Zend Framework 的重写规则：

```
if ($request_uri ~* "^/pay(.*)")
{
```

```
   set $var_pay_public '1';
}
if ($request_uri ~* ".*\.(js|ico|gif|jpg|png|css)$")
{
   set $var_pay_public '0';
}
if ($var_pay_public ~ '1')
{
   rewrite ^(.*)$ /pay/index.php last;
}
```

Bo-blog 开源 PHP 博客系统伪静态重写规则，如代码 7-2 所示：

代码 7-2

```
if (!-x $request_filename)
{
     rewrite ^/post/([0-9]+)/?([0-9]+)?/?([0-9]+)?/?$
       /read.php?entryid=$1&page=$2&part=$3 last;
     rewrite ^/page/([0-9]+)/([0-9]+)/?$ /index.php?mode=$1&page=$2 last;
     rewrite ^/starred/([0-9]+)/?([0-9]+)?/?$ /star.php?mode=$1&page=$2 last;
     rewrite ^/category/([^/]+)/?([0-9]+)?/?([0-9]+)?/?$
       /index.php?go=category_$1&mode=$2&page=$3 last;
     rewrite ^/archiver/([0-9]+)/([0-9]+)/?([0-9]+)?/?([0-9]+)?/?$
       /index.php?go=archive&cm=$1&cy=$2&mode=$3&page=$4 last;
     rewrite ^/date/([0-9]+)/([0-9]+)/([0-9]+)/?([0-9]+)?/?([0-9]+)?/?$
       /index.php?go=showday_$1-$2-$3&mode=$4&page=$5 last;
     rewrite ^/user/([0-9]+)/?$ /view.php?go=user_$1 last;
     rewrite ^/tags/([^/]+)/?([0-9]+)?/?([0-9]+)?/?$
       /tag.php?tag=$1&mode=$2&page=$3 last;
     rewrite ^/component/id/([0-9]+)/?$ /page.php?pageid=$1 last;
     rewrite ^/component/([^/]+)/?$ /page.php?pagealias=$1 last;
     #Force redirection for old rules
     rewrite ^/read\.php/([0-9]+)\.htm$ http://$host/post/$1/ permanent;
     rewrite ^/post/([0-9]+)\.htm$ http://$host/post/$1/ permanent;
     rewrite ^/post/([0-9]+)\_([0-9]+)\.htm$ http://$host/post/$1/$2/ permanent;
     rewrite ^/post/([0-9]+)\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/post/$1/$2/$3/ permanent;
     rewrite ^/index\_([0-9]+)\_([0-9]+)\.htm$ http://$host/page/$1/$2/ permanent;
     rewrite ^/star\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/starred/$1/$2/ permanent;
     rewrite ^/category\_([0-9]+)\.htm$ http://$host/category/$1/ permanent;
     rewrite ^/category\_([0-9]+)\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/category/$1/$2/$3/ permanent;
     rewrite ^/archive\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/archiver/$1/$2/ permanent;
     rewrite ^/archive\_([0-9]+)\_([0-9]+)\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/archiver/$1/$2/$3/$4/ permanent;
     rewrite ^/showday\_([0-9]+)\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/date/$1/$2/$3/ permanent;
     rewrite ^/showday\_([0-9]+)\_([0-9]+)\_([0-9]+)\_([0-9]+)\_([0-9]+)\.htm$
       http://$host/date/$1/$2/$3/$4/$5/ permanent;
```


www.TopSage.com

```
    #Filename alias
    rewrite ^/([a-zA-Z0-9_-]+)/?([0-9]+)?/?([0-9]+)?/?$
      /read.php?blogalias=$1&page=$2&part=$3 last;
}
```

根据 Referer 信息防盗链，代码如下：

```
location ~* \.(gif|jpg|png|swf|flv)$ {
   valid_referers none blocked www.yourdomain.com *.yourdomain.com;
   if ($invalid_referer) {
      rewrite ^/(.*) http://www.yourdomain.com/blocked.html;
   }
}
```

7.5　Nginx 与 Apache 的 Rewrite 规则实例对比

7.5.1　简单的 Nginx 与 Apache Rewrite 规则

一般情况下，简单的 Nginx 和 Apache Rewrite 规则区别不大，基本上能够完全兼容。例如：

Apache Rewrite 规则，代码如下：

```
RewriteRule ^/(mianshi|xianjing)/$ /zl/index.php?name=$1 [L]
RewriteRule ^/ceshi/$ /zl/ceshi.php [L]
RewriteRule ^/(mianshi)_([a-zA-Z]+)/$ /zl/index.php?name=$1_$2 [L]
RewriteRule ^/pingce([0-9]*)/$ /zl/pingce.php?id=$1 [L]
```

Nginx Rewrite 规则，代码如下：

```
rewrite ^/(mianshi|xianjing)/$ /zl/index.php?name=$1 last;
rewrite ^/ceshi/$ /zl/ceshi.php last;
rewrite ^/(mianshi)_([a-zA-Z]+)/$ /zl/index.php?name=$1_$2 last;
rewrite ^/pingce([0-9]*)/$ /zl/pingce.php?id=$1 last;
```

由以上示例可以看出，Apache 的 Rewrite 规则改为 Nginx 的 Rewrite 规则，其实很简单：Apache 的 RewriteRule 指令换成 Nginx 的 rewrite 指令，Apache 的[L]标记换成 Nginx 的 last 标记，中间的内容不变。

如果 Apache 的 Rewrite 规则改为 Nginx 的 Rewrite 规则后，使用 nginx -t 命令检查发现 nginx.conf 配置文件有语法错误（主要是大括号引起的），那么可以尝试给条件加上引号。例如以下的 Nginx Rewrite 规则会报语法错误：

```
rewrite ^/([0-9]{5}).html$ /x.jsp?id=$1 last;
```

加上引号就正确了：

```
rewrite "^/([0-9]{5}).html$" /x.jsp?id=$1 last;
```



Apache 与 Nginx 的 Rewrite 规则在 URL 跳转时有细微的区别：

Apache Rewrite 规则，如下：

```
RewriteRule ^/html/tagindex/([a-zA-Z]+)/.*$ /$1/ [R=301,L]
```

Nginx Rewrite 规则，如下：

```
rewrite ^/html/tagindex/([a-zA-Z]+)/.*$ http://$host/$1/ permanent;
```

以上示例中，我们注意到，Nginx Rewrite 规则的置换串中增加了“http://$host”，这是在 Nginx 中要求的。

另外，Apache 与 Nginx 的 Rewrite 规则在变量名称方面也有区别，例如：

Apache Rewrite 规则，如下：

```
RewriteRule ^/user/login/$ /user/login.php?login=1&forward=http://%{HTTP_HOST} [L]
```

Nginx Rewrite 规则，如下：

```
rewrite ^/user/login/$ /user/login.php?login=1&forward=http://$host last;
```

下面，我们来介绍 Apache 与 Nginx Rewrite 规则的一些功能相同或类似的指令、标记对应关系：

- Apache 的 RewriteCond 指令对应 Nginx 的 if 指令；
- Apache 的 RewriteRule 指令对应 Nginx 的 rewrite 指令；
- Apache 的[R]标记对应 Nginx 的 redirect 标记；
- Apache 的[P]标记对应 Nginx 的 last 标记；
- Apache 的[R,L]标记对应 Nginx 的 redirect 标记；
- Apache 的[P,L]标记对应 Nginx 的 last 标记；
- Apache 的[PT,L]标记对应 Nginx 的 las 标记；

如果要编写复杂的 Rewrite 规则，请阅读下一节的示例。

7.5.2 允许指定的域名访问本站，其他域名一律跳转

Apache Rewrite 规则，代码如下：

```
RewriteCond %{HTTP_HOST}    !^(.*?)\.aaa\.com$ [NC]
RewriteCond %{HTTP_HOST}    !^192\.168\.1\.(.*?)$
RewriteCond %{HTTP_HOST}    !^localhost$
RewriteRule ^/(.*)$ http://www.aaa.com [R,L]
```

Nginx Rewrite 规则如代码 7-3 所示：

www.TopSage.com

代码 7-3

```
if ($host ~* ^(.*?)\.aaa\.com$)
{
   set $var_tz '1';
}
if ($host ~* ^192\.168\.1\.(.*?)$)
{
   set $var_tz '1';
}
if ($host ~* ^localhost)
{
   set $var_tz '1';
}
if ($var_tz !~ '1')
{
   rewrite ^/(.*)$ http://www.aaa.com/ redirect;
}
```

允许指定的域名访问本站，其他域名一律跳转到 *http://www.aaa.com*。

7.5.3 URL 重写与反向代理同时进行

Apache Rewrite 规则，代码如下：

```
ProxyRequests Off
RewriteRule ^/news/(.*)$ http://server.domain.com/$1 [P,L]
```

Nginx Rewrite 规则，代码如下：

```
location /news/
{
   proxy_pass http://server.domain.com/;
}
```

7.5.4 指定 URL 之外的 URL 进行 Rewrite 跳转

Apache Rewrite 规则，代码如下：

```
RewriteCond %{REQUEST_URI} !^/xiaoqu/admin/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/map/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/accounts/.*
RewriteCond %{REQUEST_URI} !^/xiaoqu/ajax/.*
RewriteRule ^/xiaoqu/(.*?)/(.*?)/ /xiaoqu/$2.php?name=$1 [L]
```

Nginx Rewrite 规则如代码 7-4 所示：

代码 7-4

```
if ($request_uri ~* "^/xiaoqu/admin/.*")
```

www.TopSage.com

```
{
   set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/map/.*")
{
   set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/accounts/.*")
{
   set $var_xiaoqu_admin '1';
}
if ($request_uri ~* "^/xiaoqu/ajax/.*")
{
   set $var_xiaoqu_admin '1';
}
if ($var_xiaoqu_admin !~ '1')
{
   rewrite ^/xiaoqu/(.*?)/(.*?)/$ /xiaoqu/$2.php?name=$1 last;
}
```

7.5.5　域名前缀作为重写规则变量的示例

Apache Rewrite 规则，代码如下：

```
RewriteCond %{HTTP_HOST}    ^(.*?)\.domain\.com$
RewriteCond %{HTTP_HOST}    !^qita\.domain\.com$
RewriteCond %{DOCUMENT_ROOT}/html/zhuanti/secondmarket/%1/index.htm -f
RewriteRule ^/wu/$ /html/zhuanti/secondmarket/%1/index.htm [L]
```

Nginx Rewrite 规则如代码 7-5 所示：

代码 7-5

```
if ($host ~* ^(.*?)\.domain\.com$)
{
   set $var_wupin_city $1;
   set $var_wupin '1';
}
if ($host ~* ^qita\.domain\.com$)
{
   set $var_wupin '0';
}
if (!-f $document_root/html/zhuanti/secondmarket/$var_wupin_city/index.htm)
{
   set $var_wupin '0';
}
if ($var_wupin ~ '1')
{
   rewrite ^/wu/$ /html/zhuanti/secondmarket/$var_wupin_city/index.htm last;
}
```


www.TopSage.com

大家网
TopSage.com

第8章 Nginx模块开发

一些访问量非常大、业务逻辑简单的 Web 应用，如果采用 PHP 等解析型语言去处理，虽然可行，但是在并发能力、处理速度上将受到限制，耗费的系统资源也会较大，这就要求我们增加更多的服务器来处理这类应用。而采用 Nginx 模块来处理这类 Web 应用，在性能上将得到极大的提高，大大减少服务器的数量，并将在很大程度上节省服务器的运维成本。

要编写一个 Nginx 模块，你要熟悉 Nginx 的配置文件。Nginx 配置文件主要分成 4 部分：main（全局配置）、server（虚拟主机配置）、upstream（主要为反向代理、负载均衡相关配置）和 location（目录匹配配置），每部分包含若干个指令。main 部分的指令将影响其他所有部分；server 部分的指令主要用于指定虚拟主机域名、IP 和端口；upstream 的指令用于设置反向代理及后端服务器的负载均衡；location 部分用于匹配网页位置（例如，根目录“/”、“/images”，等等）。location 部分会继承 server 部分的指令，而 server 部分会继承 main 部分的指令；upstream 既不会继承指令也不会影响其他部分。它有自己的特殊指令，不需要在其他地方应用。

8.1 Nginx 模块概述

Nginx 的模块不能够像 Apache 那样动态添加，所有的模块都要预先编译进 Nginx 的二进制可执行文件中。

Nginx 模块有 3 种角色：

www.TopSage.com

（1）Handlers（处理模块）——用于处理 HTTP 请求并输出内容；

（2）Filters（过滤模块）——用于过滤 Handler 输出的内容；

（3）Load-balancers（负载均衡模块）——当有多于一台的后端服务器供选择时，选择一台后端服务器并将 HTTP 请求转发到该服务器。

当 Nginx 发送文件或转发请求到其他服务器时，可以用 Handlers 处理模块为其服务；当需要 Nginx 把输出压缩或在服务端加一些东西时，可以用 Filters 过滤模块；Nginx 的核心模块主要管理网络层和应用层协议，以及启动针对特定应用的一系列候选模块。

接下来，我们介绍 Nginx 模块的处理流程。

客户端发送 HTTP 请求到 Nginx 服务器→Nginx 基于配置文件中的位置选择一个合适的处理模块→负载均衡模块选择一台后端服务器（反向代理情况下）→处理模块进行处理并把输出缓冲放到第一个过滤模块上→第一个过滤模块处理后输出给第二个过滤模块→然后第二个过滤模块又到第三个过滤模块→第 N 个过滤模块→最后把处理结果发送给客户端。

模块相当于钩子，可以挂在 Nginx 的以下位置，在某些时段执行某些功能：

（1）当服务读配置文件之前；

（2）当读取在 location 和 server 部分或其他任何部分的每一个配置指令时；

（3）当 Nginx 初始化全局部分的配置时；

（4）当 Nginx 初始化主机部分（比如主机/端口）的配置时；

（5）当 Nginx 将全局部分的配置与主机部分的配置合并时；

（6）当 Nginx 初始化匹配位置部分配置时；

（7）当 Nginx 将其上层主机配置与位置部分配置合并时；

（8）当 Nginx 的主进程（master）开始时；

（9）当一个新的工作进程（worker）开始时；

（10）当一个工作进程退出时；

（11）当主进程退出时；

（12）处理 HTTP 请求时；

（13）过滤 HTTP 回复的头部时；

（14）过滤 HTTP 回复的主体时；

www.TopSage.com

（15）选择一台后端服务器时；

（16）初始化到后端服务器的请求时；

（17）重新初始化到后端服务器的请求时；

（18）处理来自后端服务器的回复时；

（19）完成与后端服务器的交互时。

8.2 Nginx 模块编写实践

众所周知，“Hello World”几乎已经变成了所有程序语言的第一个范例。“Hello World”程序指的是只在计算机屏幕上输出“Hello World”（意思为“世界，你好！”）这行字符串的计算机程序。一般来说，这是每一种计算机编程语言中最基本、最简单的程序，通常亦是初学者所编写的第一个程序。那么，在这一节，就让我们动手编写一个在浏览器输出“Hello Word”的 Nginx 模块，在实践中了解如何编写 Nginx 模块。

8.2.1 Hello World 模块编写与安装

（1）执行以下命令创建一个目录，将在该目录内编写我们的 Nginx 模块：

```
mkdir -p /opt/nginx_hello_world
cd /opt/nginx_hello_world
```

（2）开始创建 Nginx 模块所需的配置文件（名称为 config）：

```
vi /opt/nginx_hello_world/config
```

然后输入以下内容并保存退出：

```
ngx_addon_name=ngx_http_hello_world_module
HTTP_MODULES="$HTTP_MODULES ngx_http_hello_world_module"
NGX_ADDON_SRCS="$NGX_ADDON_SRCS $ngx_addon_dir/ngx_http_hello_world_module.c"
CORE_LIBS="$CORE_LIBS -lpcre"
```

（3）创建 Nginx 模块的 C 程序文件（名称格式为“ngx_http_模块名称_module.c”，在本示例中，文件名称为 ngx_http_hello_world_module.c：

```
vi /opt/nginx_hello_world/ngx_http_hello_world_module.c
```

然后输入如代码 8-1 所示内容并保存退出：

www.TopSage.com

大家网 TopSage.com

代码 8-1

```
#include <ngx_config.h>
#include <ngx_core.h>
#include <ngx_http.h>

static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf);

static ngx_command_t  ngx_http_hello_world_commands[] = {

  { ngx_string("hello_world"),
    NGX_HTTP_LOC_CONF|NGX_CONF_NOARGS,
    ngx_http_hello_world,
    0,
    0,
    NULL },

    ngx_null_command
};

static u_char  ngx_hello_world[] = "hello world";

static ngx_http_module_t  ngx_http_hello_world_module_ctx = {
  NULL,                          /* preconfiguration */
  NULL,                          /* postconfiguration */

  NULL,                          /* create main configuration */
  NULL,                          /* init main configuration */

  NULL,                          /* create server configuration */
  NULL,                          /* merge server configuration */

  NULL,                          /* create location configuration */
  NULL                           /* merge location configuration */
};

ngx_module_t ngx_http_hello_world_module = {
  NGX_MODULE_V1,
  &ngx_http_hello_world_module_ctx, /* module context */
  ngx_http_hello_world_commands,    /* module directives */
  NGX_HTTP_MODULE,                  /* module type */
  NULL,                          /* init master */
  NULL,                          /* init module */
  NULL,                          /* init process */
  NULL,                          /* init thread */
  NULL,                          /* exit thread */
  NULL,                          /* exit process */
  NULL,                          /* exit master */
  NGX_MODULE_V1_PADDING
};

static ngx_int_t ngx_http_hello_world_handler(ngx_http_request_t *r)
```


www.TopSage.com

```
{
  ngx_buf_t    *b;
  ngx_chain_t  out;

  r->headers_out.content_type.len = sizeof("text/plain") - 1;
  r->headers_out.content_type.data = (u_char *) "text/plain";

  b = ngx_pcalloc(r->pool, sizeof(ngx_buf_t));

  out.buf = b;
  out.next = NULL;

  b->pos = ngx_hello_world;
  b->last = ngx_hello_world + sizeof(ngx_hello_world);
  b->memory = 1;
  b->last_buf = 1;

  r->headers_out.status = NGX_HTTP_OK;
  r->headers_out.content_length_n = sizeof(ngx_hello_world);
  ngx_http_send_header(r);

  return ngx_http_output_filter(r, &out);
}

static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
  ngx_http_core_loc_conf_t  *clcf;

  clcf = ngx_http_conf_get_module_loc_conf(cf, ngx_http_core_module);
  clcf->handler = ngx_http_hello_world_handler;

  return NGX_CONF_OK;
}
```

（4）下载 Nginx 源码包，并将 hello world 模块编译到其中，如代码 8-2 所示：

代码 8-2

```
wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.9.tar.gz
tar zxvf pcre-7.9.tar.gz
cd pcre-7.9/
./configure
make && make install
cd ../

wget http://sysoev.ru/nginx/nginx-0.8.14.tar.gz
tar zxvf nginx-0.8.14.tar.gz
cd nginx-0.8.14/
./configure --prefix=/usr/local/nginx --add-module=/opt/nginx_hello_world
make
make install
```


www.TopSage.com

（5）配置 nginx.conf，在 server 部分增加以下内容：

```
location = /hello {
  hello_world;
}
```

（6）启动 Nginx，用浏览器访问 *http://localhost/hello*，就可以看到编写的 Nginx Hello World 模块输出的文字“Hello World”。

8.2.2 Hello World 模块分析

（1）在 ngx_http_hello_world_module.c 代码中，ngx_command_t 函数用于定义包含模块指令的静态数组 ngx_http_hello_world_commands，如代码 8-3 所示。

代码 8-3

```
static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf);

static ngx_command_t  ngx_http_hello_world_commands[] = {

  { ngx_string("hello_world"),
    NGX_HTTP_LOC_CONF|NGX_CONF_NOARGS,
    ngx_http_hello_world,
    0,
    0,
    NULL },

    ngx_null_command
};
```

在数组中，第一项参数 ngx_string("hello_world")为指令名称字符串，不能含有空格，数据类型是 ngx_str_t，经常用来进行字符串实例化。

注意：ngx_str_t 结构体由包含有字符串的 data 成员和表示字符串长度的 len 成员组成。Nginx 用这个数据类型来存放字符串。

第二项参数用来设置指令在配置文件位置的哪一部分使用是合法的，可选值如下，多个选项以“|”隔开：

NGX_HTTP_MAIN_CONF——指令出现在全局配置部分是合法的；

NGX_HTTP_SRV_CONF——指令出现在 server 主机配置部分是合法的；

NGX_HTTP_LOC_CONF——指令出现在 location 配置部分是合法的；

NGX_HTTP_UPS_CONF——指令出现在 upstream 配置部分是合法的；

NGX_CONF_NOARGS——指令没有参数；

www.TopSage.com

大家网 TopSage.com

NGX_CONF_TAKE1——指令读入 1 个参数；

NGX_CONF_TAKE2——指令读入 2 个参数；

……

NGX_CONF_TAKE7——指令读入 7 个参数；

NGX_CONF_FLAG——指令读入 1 个布尔型数据；

NGX_CONF_1MORE——指令至少读入 1 个参数；

NGX_CONF_2MORE——指令至少读入 2 个参数。

第三项参数 ngx_http_hello_world 是一个回调函数。该回调函数有 3 个参数：

ngx_conf_t *cf——指向 ngx_conf_t 结构体的指针，包含从指令后面传过来的参数。

ngx_command_t *cmd——指向当前 ngx_command_t 结构体的指针。

void *conf——指向自定义模块配置结构体的指针。

第四项参数用于告诉 Nginx 是把要保持的值放在全局配置部分、server 主机配置部分还是 location 位置配置部分（使用 NGX_HTTP_MAIN_CONF_OFFSET、NGX_HTTP_SRV_CONF_OFFSET、NGX_HTTP_LOC_CONF_OFFSET）。

第五项参数用于设置指令的值保存在结构体的哪个位置。第六项参数一般为 NULL。这样，6 个参数都设置完毕，此数组在读入 ngx_null_command 后停止。

（2）接下来的 static u_char ngx_hello_world[] = "hello world";语句，则定义了用于输出给客户端浏览器的字符串内容。

（3）ngx_http_module_t 用来定义结构体 ngx_http_hello_world_module_ctx，如代码 8-4 所示。

代码 8-4

```
static ngx_http_module_t  ngx_http_hello_world_module_ctx = {
  NULL,                          /* preconfiguration */
  NULL,                          /* postconfiguration */

  NULL,                          /* create main configuration */
  NULL,                          /* init main configuration */

  NULL,                          /* create server configuration */
  NULL,                          /* merge server configuration */

  NULL,                          /* create location configuration */
  NULL                           /* merge location configuration */
};
```

静态 ngx_http_module_t 结构体，包含很多函数引用，用来创建 3 个部分的配置和合并配置。一般结构体命名为 ngx_http_<module name>_module_ctx。这些函数引用包括：

preconfiguration——在读入配置前调用；

postconfiguration——在读入配置后调用；

create main configuration——在创建全局部分配置时调用（比如，用来分配空间和设置默认值）；

init main configuration——在初始化全局部分的配置时调用（比如，把原来的默认值用 nginx.conf 读到的值覆盖）；

create server configuration——在创建虚拟主机部分的配置时调用；

merge server configuration——与全局部分配置合并时调用；

create location configuration——创建位置部分的配置时调用；

merge location configuration——与主机部分配置合并时调用。

这些函数参数不同，依赖于它们的功能。代码 8-5 是这个结构体的定义，摘自 http/ngx_http_config.h，你可以看到属性各不相同的回调函数：

代码 8-5

```
typedef struct {
   ngx_int_t   (*preconfiguration)(ngx_conf_t *cf);
   ngx_int_t   (*postconfiguration)(ngx_conf_t *cf);
   void       *(*create_main_conf)(ngx_conf_t *cf);
   char       *(*init_main_conf)(ngx_conf_t *cf, void *conf);
   void       *(*create_srv_conf)(ngx_conf_t *cf);
   char       *(*merge_srv_conf)(ngx_conf_t *cf, void *prev, void *conf);
   void       *(*create_loc_conf)(ngx_conf_t *cf);
   char       *(*merge_loc_conf)(ngx_conf_t *cf, void *prev, void *conf);
}
```

如果不用某些函数，可以设定为 NULL，Nginx 会剔除它。

（4）ngx_module_t 定义结构体 ngx_http_hello_world_module，如代码 8-6 所示。

代码 8-6

```
ngx_module_t ngx_http_hello_world_module = {
  NGX_MODULE_V1,
  &ngx_http_hello_world_module_ctx, /* module context */
  ngx_http_hello_world_commands,   /* module directives */
  NGX_HTTP_MODULE,              /* module type */
  NULL,                      /* init master */
  NULL,                      /* init module */
  NULL,                      /* init process */
```


www.TopSage.com

```
    NULL,                          /* init thread */
    NULL,                          /* exit thread */
    NULL,                          /* exit process */
    NULL,                          /* exit master */
    NGX_MODULE_V1_PADDING
};
```

这个结构体变量命名为 ngx_http_<module name>_module。它包含有模块的主要内容和指令的执行部分，也有一些回调函数（退出线程、退出进程，等等）。这个模块的定义是把数据处理关联到特定模块的关键。

（5）回调函数 ngx_http_hello_world，分为两步，第一步获得这个 Location 位置配置的"核心"结构体，然后为它分配一个处理函数 ngx_http_hello_world_handler，如代码 8-7 所示。

代码 8-7

```
static char *ngx_http_hello_world(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
  ngx_http_core_loc_conf_t  *clcf;

  clcf = ngx_http_conf_get_module_loc_conf(cf, ngx_http_core_module);
  clcf->handler = ngx_http_hello_world_handler;

  return NGX_CONF_OK;
}
```

（6）处理函数 ngx_http_hello_world_handler，也是 Hello World 模块的核心部分。参数 ngx_http_request_t *r，可以让我们访问到客户端的头部和不久要发送的回复头部，包含两个成员：r->headers_in 和 r->headers_out。

b->pos = ngx_hello_world;即为要输出的内容，ngx_hello_world 的内容就是之前通过 static u_char ngx_hello_world[] = "hello world";语句定义的用于输出给客户端浏览器的字符串"hello world"，如代码 8-8 所示。

代码 8-8

```
static ngx_int_t ngx_http_hello_world_handler(ngx_http_request_t *r)
{
  ngx_buf_t    *b;
  ngx_chain_t   out;

  r->headers_out.content_type.len = sizeof("text/plain") - 1;
  r->headers_out.content_type.data = (u_char *) "text/plain";

  b = ngx_pcalloc(r->pool, sizeof(ngx_buf_t));

  out.buf = b;
  out.next = NULL;
```

www.TopSage.com




```
  b->pos = ngx_hello_world;
  b->last = ngx_hello_world + sizeof(ngx_hello_world);
  b->memory = 1;
  b->last_buf = 1;

  r->headers_out.status = NGX_HTTP_OK;
  r->headers_out.content_length_n = sizeof(ngx_hello_world);
  ngx_http_send_header(r);

  return ngx_http_output_filter(r, &out);
}
```

通过以上内容，相信您已经掌握了如何编写一个简单的 Nginx 模块。Nginx 的模块开发涉及的内容非常广，编写更为复杂的 Nginx 模块，不是本书的重点所在。如果您感兴趣，可以参阅 Evan Miller 编写的《Evan Miller's Guide to Nginx Module Development》(*http://www.evanmiller.org/nginx-modules-guide.html*) 获取更多的信息。

www.TopSage.com

第 9 章

Nginx 的 Web 缓存服务与新浪网的开源 NCACHE 模块

9.1 什么是 Web 缓存

Web 缓存位于内容源 Web 服务器和客户端之间，当用户访问一个 URL 时，Web 缓存服务器会去后端 Web 源服务器取回要输出的内容，然后，当下一个请求到来时，如果访问的是相同的 URL，Web 缓存服务器直接输出内容给客户端，而不是向源服务器再次发送请求。Web 缓存降低了内容源 Web 服务器、数据库的负载，减少了网络延迟，提高了用户访问的响应速度，增强了用户体验。

Web 缓存服务器中，最著名的要数 Squid Cache（简称为 Squid），已经在大多数网站中使用。Squid 是一个流行的自由软件（GNU 通用公共许可证）的代理服务器和 Web 缓存服务器。Squid 有广泛的用途，从作为网页服务器的前置 cache 服务器缓存相关请求来提高 Web 服务器的速度，到为一组人共享网络资源而缓存万维网、域名系统和其他网络搜索，到通过过滤流量帮助网络安全，到局域网通过代理上网。Squid 主要设计用于在 Unix 一类系统运行。

现在，新版本 Nginx 的 proxy_cache 指令开始支持 Web 缓存服务，另外，新浪网为 Nginx 开发的 NCACHE 模块，也能够支持 Web 缓存服务，解决了 Squid 不能充分利用多核 CPU 的局限，速度比 Squid 更快。

www.TopSage.com

9.2 Nginx 的 Web 缓存服务

Nginx 从 0.7.48 版开始，支持了类似 Squid 的缓存功能。这个缓存是把 URL 及相关组合当作 Key，用 md5 算法对 Key 进行哈希，得到硬盘上对应的哈希目录路径，从而将缓存内容保存在该目录内。它可以支持任意 URL 链接，同时也支持 404/301/302 这样的非 200 状态码。虽然目前官方的 Nginx Web 缓存服务只能为指定 URL 或状态码设置过期时间，不支持类似 Squid 的 PURGE 指令，手动清除指定缓存页面，但是，通过一个第三方的 ngx_cache_purge 模块，可以清除指定 URL 的缓存。

金山逍遥网已经在生产环境使用 Nginx 的 proxy_cache 缓存功能多月，十分稳定，速度不逊于 Squid。在功能上，Nginx 已经具备 Squid 所拥有的 Web 缓存加速功能、清除指定 URL 缓存的功能。而在性能上，Nginx 对多核 CPU 的利用，胜过 Squid 不少。另外，在反向代理、负载均衡、健康检查、后端服务器故障转移、重写、易用性上，Nginx 也比 Squid 强大很多。这使得一台 Nginx 可以同时作为“负载均衡服务器”与“Web 缓存服务器”来使用。

Nginx 的 Web 缓存服务主要由 proxy_cache 相关指令集和 fastcgi 相关指令集构成，前者用于反向代理时，对后端内容源服务器进行缓存，后者主要用于对 FastCGI 的动态程序进行缓存。两者的功能基本上一样。

9.2.1 proxy_cache 相关指令集

1. proxy_cache 指令

语法：proxy_cache zone_name;

默认值：None

使用环境：http, server, location

该指令用于设置哪个缓存区将被使用，zone_name 的值为 proxy_cache_path 指令创建的缓存区名称。

2. proxy_cache_path 指令

语法：proxy_cache_path path [levels=number] keys_zone=zone_name:zone_size [inactive=time] [max_size=size];

默认值：None

使用环境：http

该指令用于设置缓存文件的存放路径。示例如下：

www.TopSage.com

大家网 TopSage.com

```
proxy_cache_path  /data0/proxy_cache_dir  levels=1:2  keys_zone=cache_one:
 500m inactive=1d max_size=30g;
```

注意该指令只能在 http 标签内配置，levels 指定该缓存空间有两层 hash 目录，第一层目录为 1 个字母，第二层为 2 个字母，保存的文件名会类似/data0/proxy_cache_dir/c/29/b7f54b2df7773722d-382f4809d65029c；keys_zone 参数用来为这个缓存区起名，500m 指内存缓存空间大小为 500MB；inactive 的 1d 指如果缓存数据在 1 天内没有被访问，将被删除；max_size 的 30g 是指硬盘缓存空间为 30GB。

3. proxy_cache_methods 指令

语法：proxy_cache_methods [GET HEAD POST];

默认值：proxy_cache_methods GET HEAD;

使用环境：http, server, location

该指令用于设置缓存哪些 HTTP 方法，默认缓存 HTTP GET/HEAD 方法，不缓存 HTTP POST 方法。

4. proxy_cache_min_uses 指令

语法：proxy_cache_min_uses the_number;

默认值：proxy_cache_min_uses 1;

使用环境：http, server, location

该指令用于设置缓存的最小使用次数，默认值为 1。

5. proxy_cache_valid 指令

语法：proxy_cache_valid reply_code [reply_code ...] time;

默认值：None

使用环境：http, server, location

该指令用于对不同返回状态码的 URL 设置不同的缓存时间，例如：

```
proxy_cache_valid  200 302  10m;
proxy_cache_valid  404      1m;
```

设置 200、302 状态的 URL 缓存 10 分钟，404 状态的 URL 缓存 1 分钟。

```
proxy_cache_valid 5m;
```

如果不指定状态码，直接指定缓存时间，则只有 200、301、302 状态的 URL 缓存 5 分钟。

```
proxy_cache_valid 200 302 10m;
proxy_cache_valid 301 1h;
```

www.TopSage.com

```
proxy_cache_valid  any 1m;
```

对没有单独设置的状态码，全部设置缓存时间为 1 分钟。

6. proxy_cache_key 指令

语法：proxy_cache_key　line;

默认值：None

使用环境：http, server, location

该指令用来设置 Web 缓存的 Key 值，Nginx 根据 Key 值 md5 哈希存储缓存。一般根据$host（域名）、$request_uri（请求的路径）等变量组合成 proxy_cache_key。例如：

```
proxy_cache_key "$host:$server_port$uri$is_args$args";
```

9.2.2 proxy_cache 完整示例

上一节中，我们已经介绍了 proxy_cache 的相关指令集，现在来看一个 proxy_cache 的完整示例。

（1）首先，我们要按照以下步骤，把第三方的 ngx_cache_purge 模块编译安装到 Nginx 中，用来清除指定 URL 的缓存，示例如代码 9-1 所示。

代码 9-1

```
ulimit -SHn 65535
wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.00.tar.gz
tar zxvf pcre-8.00.tar.gz
cd pcre-8.00/
./configure
make && make install
cd ../

wget http://labs.frickle.com/files/ngx_cache_purge-1.0.tar.gz
tar zxvf ngx_cache_purge-1.0.tar.gz

wget http://nginx.org/download/nginx-0.8.32.tar.gz
tar zxvf nginx-0.8.32.tar.gz
cd nginx-0.8.32/
./configure --user=www --group=www --add-module=../ngx_cache_purge-1.0
--prefix=/usr/local/webserver/nginx --with-http_stub_status_module
--with-http_ssl_module
make && make install
cd ../
```

（2）然后，在同一分区下创建两个缓存目录，分别供 proxy_temp_path、proxy_cache_path 指令设置缓存路径。

www.TopSage.com

注：两个指定设置的缓存路径必须为同一磁盘分区，不能跨分区。

```
mkdir -p /data0/proxy_temp_path
mkdir -p /data0/proxy_cache_path
```

（3）Nginx 配置文件（nginx.conf）：对扩展名为 gif、jpg、jpeg、png、bmp、swf、js、css 的图片、Flash、JavaScript、CSS 文件开启 Web 缓存，其他文件不缓存，示例如代码 9-2 所示。

代码 9-2

```
user  www www;

worker_processes 8;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 51200;

events
{
 use epoll;
 worker_connections 51200;
}

http
{
 include       mime.types;
 default_type  application/octet-stream;

 #charset  utf-8;

 server_names_hash_bucket_size 128;
 client_header_buffer_size 32k;
 large_client_header_buffers 4 32k;

 sendfile on;
 #tcp_nopush     on;

 keepalive_timeout 30;

 tcp_nodelay on;

 #注: proxy_temp_path 和 proxy_cache_path 指定的路径必须在同一分区
 proxy_temp_path /data0/proxy_temp_path;
 #设置 Web 缓存区名称为 cache_one，内存缓存空间大小为 500MB，自动清除超过 1 天没有被访问的缓存数据，
硬盘缓存空间大小为 30GB。
 proxy_cache_path  /data0/proxy_cache_path  levels=1:2   keys_zone=cache_one:200m
inactive=1d max_size=30g;
```


www.TopSage.com

大家网 TopSage.com

```
upstream my_server_pool {
  server   192.168.1.2:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.3:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.4:80 weight=1 max_fails=2 fail_timeout=30s;
}

server
{
  listen      80;
  server_name  my.domain.com;

  location /
  {
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
      proxy_pass http://my_server_pool;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|js|css)$
  {
      #使用 Web 缓存区 cache_one
      proxy_cache cache_one;

      #对不同 HTTP 状态码缓存设置不同的缓存时间
      proxy_cache_valid  200 304 12h;
      proxy_cache_valid  301 302 1m;
      proxy_cache_valid  any 1m;

      #设置 Web 缓存的 Key 值，Nginx 根据 Key 值 md5 哈希存储缓存，这里根据“域名、URI、参数”组
       合成 Key。
      proxy_cache_key $host$uri$is_args$args;

      #反向代理，访问后端内容源服务器
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
      proxy_pass http://my_server_pool;
  }

  #用于清除缓存，假设一个 URL 为 http:// my.domain.com/test.gif，通过访问
   http://my.domain.com/purge/test.gif 可以清除该 URL 的缓存。
  location ~ /purge(/.*)
  {
      #设置只允许指定的 IP 或 IP 段才可以清除 URL 缓存。
      allow          127.0.0.1;
      allow          192.168.0.0/16;
      deny          all;
      proxy_cache_purge    cache_one    $host$1$is_args$args;
  }

  access_log  off;
}
}
```


www.TopSage.com

9.2.3 fastcgi_cache 相关指令集

1. fastcgi_cache 指令

语法：fastcgi_cache zone_name;

默认值：off

使用环境：http, server, location

该指令用于设置哪个缓存区将被使用，zone_name 的值为 fastcgi_cache_path 指令创建的缓存区名称。

2. fastcgi_cache_path 指令

语法：fastcgi_cache_path path [levels=number] keys_zone=zone_name:zone_size [inactive=time] [max_size=size];

默认值：None

使用环境：http

该指令用于设置缓存文件的存放路径。示例如下：

```
fastcgi_cache_path /data0/fastcgi_cache_dir levels=1:2  keys_zone=cache_one:500m
inactive=1d max_size=30g;
```

注意该指令只能在 http 标签内配置，levels 指定该缓存空间有两层 hash 目录，第一层目录为 1 个字母，第二层为 2 个字母，保存的文件名会类似/data0/fastcgi_cache_dir/c/29/b7f54b2df-7773722d382f4809d65029c；keys_zone 参数用来为这个缓存区起名，500m 指内存缓存空间大小为 500MB；inactive 的 1d 指如果缓存数据在 1 天内没有被访问，将被删除；max_size 的 30g 是指硬盘缓存空间为 30GB。

3. fastcgi_cache_methods 指令

语法：fastcgi_cache_methods [GET HEAD POST];

默认值：fastcgi_cache_methods GET HEAD;

使用环境：http, server, location

该指令用于设置缓存哪些 HTTP 方法，默认缓存 HTTP GET/HEAD 方法，不缓存 HTTP POST 方法。

4. fastcgi_cache_min_uses 指令

语法：fastcgi_cache_min_uses the_number;

www.TopSage.com

默认值：fastcgi_cache_min_uses 1;

使用环境：http, server, location

该指令用于设置缓存的最小使用次数，默认值为 1。

5. fastcgi_cache_valid 指令

语法：fastcgi_cache_valid reply_code [reply_code ...] time;

默认值：None

使用环境：http, server, location

该指令用于对不同返回状态码的 URL 设置不同的缓存时间，例如：

```
fastcgi_cache_valid  200 302  10m;
fastcgi_cache_valid  404     1m;
```

设置 200、302 状态的 URL 缓存 10 分钟，404 状态的 URL 缓存 1 分钟。

```
fastcgi_cache_valid 5m;
```

如果不指定状态码，直接指定缓存时间，则只有 200、301、302 状态的 URL 缓存 5 分钟。

```
fastcgi_cache_valid  200 302 10m;
fastcgi_cache_valid  301 1h;
fastcgi_cache_valid  any 1m;
```

对没有单独设置的状态码，全部设置缓存时间为 1 分钟。

6. fastcgi_cache_key 指令

语法：fastcgi_cache_key line;

默认值：None

使用环境：http, server, location

该指令用来设置 Web 缓存的 Key 值，Nginx 根据 Key 值 md5 哈希存储缓存。一般根据 FastCGI 服务器的地址和端口、$request_uri（请求的路径）等变量组合成 fastcgi_cache_key。例如：

```
fastcgi_cache_key 127.0.0.1:9000$request_uri;
```

9.2.4 fastcgi_cache 完整示例

上一节中，我们已经介绍了 fastcgi_cache 的相关指令集，现在来看一个 fastcgi_cache 的完整示例：

（1）首先，在同一分区下创建两个缓存目录，分别供 fastcgi_temp_path、fastcgi_cache_path 指令设置缓存路径。

www.TopSage.com

大家网 TopSage.com

注：两个指定设置的缓存路径必须为同一磁盘分区，不能跨分区。

```
mkdir -p /data0/fastcgi_temp_path
mkdir -p /data0/fastcgi_cache_path
```

（2）Nginx 配置文件（nginx.conf）：对扩展名为 gif、jpg、jpeg、png、bmp、swf、js、css 的图片、Flash、JavaScript、CSS 文件开启 Web 缓存，其他文件不缓存，示例如代码 9-3 所示。

代码 9-3

```
user  www www;

worker_processes 8;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 51200;

events
{
 use epoll;
 worker_connections 51200;
}

http
{
 include       mime.types;
 default_type  application/octet-stream;

 #charset  utf-8;

 server_names_hash_bucket_size 128;
 client_header_buffer_size 32k;
 large_client_header_buffers 4 32k;

 sendfile on;
 #tcp_nopush     on;

 keepalive_timeout 30;

 tcp_nodelay on;

 #注: fastcgi_temp_path 和 fastcgi_cache_path 指定的路径必须在同一分区
 fastcgi_temp_path /data0/fastcgi_temp_path;
 #设置 Web 缓存区名称为 cache_one,内存缓存空间大小为 500MB,自动清除超过 1 天没有被访问的缓存数据,
  硬盘缓存空间大小为 30GB。
 fastcgi_cache_path  /data0/fastcgi_cache_path  levels=1:2
keys_zone=cache_one:200m inactive=1d max_size=30g;
```


www.TopSage.com

```
upstream my_server_pool {
  server   192.168.1.2:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.3:80 weight=1 max_fails=2 fail_timeout=30s;
  server   192.168.1.4:80 weight=1 max_fails=2 fail_timeout=30s;
}

server
{
  listen      80;
  server_name  my.domain.com;
  root /data0/htdocs;

  location ~ .*\.(php|php5)$
  {
      #使用 Web 缓存区 cache_one
      fastcgi_cache cache_one;

      #对不同 HTTP 状态码缓存设置不同的缓存时间
      fastcgi_cache_valid  200 10m;
      fastcgi_cache_valid  301 302 1h;
      fastcgi_cache_valid  any 1m;

      #设置 Web 缓存的 Key 值，Nginx 根据 Key 值 md5 哈希存储缓存，这里根据“FastCGI 服务器的 IP、
       端口、请求的 URI”组合成 Key。
      fastcgi_cache_key 127.0.0.1:9000$request_uri;

      #FastCGI 服务器
      fastcgi_pass  127.0.0.1:9000;
      fastcgi_index index.php;
      include fcgi.conf;
  }

  access_log  off;
 }
}
```

9.3　新浪网开源软件项目——基于 Nginx 的 NCACHE 网页缓存系统

NCACHE 是基于 Nginx 的 Web 服务器模型构建起来的缓存系统，是新浪公司的开源产品。NCACHE 最早的时候是作为 Nginx 的一个 HTTP 模块进行开发的，因为当时希望获得更好的兼容性和可扩展性，作为独立模块，可以被更好地推广和使用，安装也会很方便。但后来发现随着代码量的增加、功能的扩充，Nginx 的原有模块框架已经不能很好地满足要求了，因此，我们提取了 Nginx 的内核代码，并把 CACHE 部分嵌入其中，形成了今天的 NCACHE。

www.TopSage.com

NCACHE 本身功能并不强大，且不具备像 SQUID 般完善的功能和开发框架，甚至不能支持 RFC 中关于 CACHE 部分的描述。NCACHE 完全是一套定制化的产品，可以满足要快速部署、简单易用、大并发量、大存储量的需求，它不需要复杂的配置，不需要冗余的复杂代码，并使用最先进的技术组合。

NCACHE 2.0 版本，是作为一个完整的 Nginx 模块进行发布和使用的，从原有的 NCACHE 内核中进行了剥离，更方便开发者的安装和配置。

NCACHE 3.0 版本，相对于 2.0 版本有了很大的改进，对文件的缓存不再使用传统的目录模式，而是通过 MMAP 一个大文件，在其中以页分配的形式存储缓存数据，由操作系统来负责决定哪些数据应该留在内存里，这与 VARNISH 缓存的原理是一致的，大大提高了 IO 性能，目前该版本只支持 64 位 Linux 和 FREEBSD 系统。

9.3.1 NCACHE 模块的安装

NCACHE 需要在编译 Nginx 时，将自己编译进 Nginx。NCACHE 模块最新的版本为 ncache-3.1.64，目前该版本只支持 64 位 Linux 和 FREEBSD 操作系统，ncache-3.1.64 能够兼容 nginx-0.6.x 系列版本，目前还不兼容 nginx-0.7.x、nginx-0.8.x 版本。

下面，我们将 ncache-3.1.64 版本编译进 nginx-0.6.39，如代码 9-4 所示：

代码 9-4

```
mkdir -p /data0/software
cd /data0/software
wget http://ncache.googlecode.com/files/ncache-3.1_64_linux.tar.gz
tar zxvf ncache-3.1_64_linux.tar.gz
wget http://sysoev.ru/nginx/nginx-0.6.39.tar.gz
tar zxvf nginx-0.6.39.tar.gz
cd nginx-0.6.39/
./configure --prefix=/usr/local/webserver/ncache --add-module=../ncache --user=www
--group=www --with-http_stub_status_module --with-http_ssl_module
make && make install
```

9.3.2 NCACHE 配置文件编写

创建配置文件/usr/local/webserver/ncache/conf/nginx.Conf，如代码 9-5 所示：

代码 9-5

```
user www www;

worker_processes 4;
```


www.TopSage.com



```
worker_rlimit_nofile 20480;

error_log  logs/error.log;

events
{
      use epoll;

      worker_connections 81920;
}

http
{
      keepalive_timeout 10;

      ncache_max_size 24;

      proxy_buffering off;

      ncache_dir /data1/cache_file  60;#the file for storage,60G
      ncache_dir /data2/cache_file  60;
      hash_index_dir /data1/ncache_index;#v3.1,you need to define the index file
        position
      auto_delete_file on;#enable the auto delete file
      ncache_ignore_client_no_cache on;

      upstream backend
      {
         server 10.0.0.1;
      }

      sendfile on;

      send_timeout 10;

      client_header_timeout 10;


      tcp_nodelay on;

      log_format main '$proxy_add_x_forwarded_for - $remote_user [$time_local]'
                      '"$request" $status $bytes_sent '
                      '"$http_referer" "$http_user_agent" $remote_addr';

      include "mime.types";

      default_type text/html;

      server
      {
            server_name  .blog.sina.com.cn;
```


www.TopSage.com

```
listen *:80;

set $xvia "blog.sina.com.cn";

set $proxy_add_agent "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
 NginxCache)";
access_log logs/blog.sina.com.cn-access_log main;

location /
{

       if ($request_method ~ "PURGE")
       {
                     rewrite (.*) /PURGE$1 last;
       }


       ncache_http_cache;

       error_page  404 = /fetch$request_uri;

       add_header     Sina-Cache   $xvia;

}

location /ncache_state
{
   ncache_state;
}


location /fetch
{
   internal;
   proxy_pass http://backend;
   add_header     Sina-Cache   $xvia;
   proxy_hide_header   User-Agent;
   proxy_set_header    User-Agent $proxy_add_agent;
   proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
}


location /PURGE/
{
   access_log logs/purge.blog.sina.com.cn-access_log main;
   internal;
   allow 10.55.37.0/24;
   allow 10.69.3.0/24;
   allow 10.49.10.0/24;
   deny   all;
   ncache_purge;
}
```

```
        }
}
```

9.3.3 NCACHE 的管理维护

1. 运行 NCACHE

```
/usr/local/webserver/ncache/sbin/nginx
```

2. 停止 NCACHE

```
killall -9 nginx
```

3. 清除缓存的 Shell 脚本

```
#! /bin/sh
kill -9 `ps auwx|grep nginx|awk '{print $2}'`
killall -9 nginx
rm ../logs/ncache_index
echo "" > ../logs/error.log
rm /data0/ncache_file
rm /data1/ncache_file
rm /data2/ncache_file
rm /data3/ncache_file
```

4. 查看 NCACHE 状态

```
curl "http://127.0.0.1/ncache_state"
```

9.3.4 NCACHE 后端内容源服务器设置

（1）必须用 HTTP Header 头“Cache-Control: max-age=秒数”来控制缓存时间，如果不指定将不缓存。

（2）后端内容源服务器发送 NCACHE 的 HTTP 数据，必须带有“Content-Length”Header 头。

（3）NCACHE 的缓存时间以分钟为单位，会将所有 max-age=的秒数值转换为分钟，如果 max-age 小于 1 分钟，NCACHE 会将缓存时间设置成 1 分钟。

（4）自动删除缓存文件进程，会在每天的凌晨 2 点删除大约 20%的不活动缓存数据。

 www.TopSage.com

第 10 章

Nginx 在国内知名网站中的应用案例

Nginx 在国内知名网站中的应用案例主要可分为三类：Nginx 反向代理与负载均衡类网站应用案例、Nginx+PHP/JSP 类网站应用案例、Nginx 静态内容 Web 服务器应用案例。下面，我们找了一些截至 2009 年 9 月，使用 Nginx 作为 Web 服务器的网站代表：

1. 使用 Nginx 运行 PHP（FastCGI）、JSP 程序的网站

新浪播客（*http://v.sina.com.cn/*）：nginx/0.7.62 + PHP

金山逍遥网（*http://www.xoyo.com/*）：nginx/0.8.15 + PHP 5.2.9

金山爱词霸（*http://www.iciba.com/*）：nginx/0.6.32 + PHP

六间房视频（*http://www.6.cn/*）：nginx/0.7.21 + PHP

Discuz!官方论坛（*http://www.discuz.net/*）：nginx/0.7.59 + PHP/5.2.10

赶集网（*http://www.ganji.com/*）：nginx/0.7.62 + PHP/5.2.8

搜狐通行证（*http://passport.sohu.com/*）：nginx/0.6.37 + JSP

网易博客（*http://blog.163.com/*）：nginx/0.7.59 + JSP

人人网（原校内网）（*http://renren.com/*）：nginx/0.6.32 + JSP

2. 使用 Nginx 作反向代理、规则过滤的网站

新浪播客接口服务器（*http://interface.video.sina.com.cn/*）：nginx/0.5.35

www.TopSage.com



新浪博客（*http://blog.sina.com.cn/*）：nginx/0.7.62

YUPOO 相册（*http://www.yupoo.com/*）：nginx/0.5.35

金山游戏用户中心（*http://my.xoyo.com/*）：nginx/0.8.15

豆瓣（*http://www.douban.com/*）：nginx

3. 使用 Nginx 运行静态 HTML 页、图片、FLV 视频的网站

网易新闻（*http://news.163.com*）：nginx/0.6.36

酷 6 网（*http://www.ku6.com/*）视频服务器：nginx/0.5.36

迅雷看看（*http://www.xunlei.com/*）：nginx/0.6.31

新华网 RSS 订阅频道（*http://rss.xinhuanet.com/*）：nginx

腾讯网（*http://www.qq.com/*）：nginx/0.6.39

开心网图片服务器（*http://img1.kaixin001.com.cn/*）：nginx

无论是新浪、搜狐、网易、腾讯等门户网站，还是目前红红火火的 Web 2.0 网站，都已经有使用 Nginx 作为某些项目的 Web 服务器案例。Nginx 大有取代 Apache 之势。下面，我们就来结合实际案例，介绍一下 Nginx 在国内知名网站中的应用。

10.1 Nginx 反向代理与负载均衡类网站应用案例

10.1.1 Nginx 负载均衡在新浪播客中的应用

2008 年的新浪播客（v.sina.com.cn、you.video.sina.com.cn）由静态服务器集群和动态服务器集群两部分组成，分别采用不同的域名。静态服务器集群即我们在浏览器地址栏经常能看到 v.sina.com.cn 和 you.video.sina.com.cn 域名，采用 Squid 做前端缓存，服务器分布在全国各地机房。动态服务器集群采用 interface.video.sina.com.cn 域名，也称接口服务器，主要用来实时显示播放数、记录播放日志、为 Flash 视频播放器提供数据、与新浪内部产品、外部合作产品交互。

接口服务器最上层采用 F5 BIG-IP 硬件四/七层负载均衡交换机，对 4 台 Nginx 反向代理服务器进行四层负载均衡，由这 4 台 Nginx 服务器判断 URL，进行分组，对后端的 3 组 Web 服务器进行七层负载均衡。F5 BIG-IP 也支持七层负载均衡，但是，由于一对 F5 BIG-IP 要服务新浪的多个产品，七层交换需要耗费 F5 BIG-IP 不少的 CPU 资源，而 F5 BIG-IP 四层交换有专门的硬件芯片来处理，耗费的资源较少，所以，在新浪，F5 BIG-IP 一般只用四层负载均衡。

www.TopSage.com

F5 BIG-IP 后端的 3 组 Web 服务器，配置不一样，第 1 组为内存密集型服务器，技术类型主要是 PHP+Memcached 服务；第 2 组为 CPU 密集型服务器，主要耗费的是 CPU 资源；第 3 组为磁盘密集型服务器，为记录日志等操作，要求磁盘空间大、磁盘转速高。

代码 10-1 是新浪播客接口服务器的 Nginx 负载均衡配置，提供按 URL 分组服务、负载均衡服务：

代码 10-1

```
user  www www;

worker_processes 10;

error_log  /data1/logs/nginx_error.log  crit;

pid      /tmp/nginx.pid;

worker_rlimit_nofile 51200;

events
{
     use epoll;

     worker_connections 51200;
}

http
{
     include      conf/mime.types;
     default_type  application/octet-stream;

     charset  gb2312;

     server_names_hash_bucket_size 128;

     keepalive_timeout 15;

     sendfile on;
     tcp_nopush on;
     tcp_nodelay on;

     #第 1 组接口机: Memcache 相关（点击数）
     upstream  count.interface.video.sina.com.cn  {
           server   xx.xx.xx.55:80;
           server   xx.xx.xx.58:80;
           server   xx.xx.xx.47:80;
     }

     #第 2 组接口机: 外部提供数据类程序
     upstream  data.interface.video.sina.com.cn  {
```

www.TopSage.com

```
        server   xx.xx.xx.59:80;
        server   xx.xx.xx.64:80;
        server   xx.xx.xx.48:80;
}

#第 3 组接口机：打日志类程序、功能相关、嵌套页面
upstream  log.interface.video.sina.com.cn  {
        server   xx.xx.xx.72:80;
        server   xx.xx.xx.49:80;
}

server
{
        listen  80;
        server_name interface.video.sina.com.cn;

        location / {
                proxy_redirect off;

                #后端的 Web 服务器可以通过 X-Forwarded-For 获取用户真实 IP
                proxy_set_header   X-Forwarded-For  $remote_addr;

                #按 URL 进行分组，第 1 组：Memcache 相关（点击数）
                if ($request_uri ~ "^\/app\/count\/")
                {
                        proxy_pass http://count.interface.video.sina.com.cn;
                }
                if ($request_uri ~ "^\/app\/online\/")
                {
                        proxy_pass http://count.interface.video.sina.com.cn;
                }
                if ($request_uri ~ "^\/interface\/user\/getLoginGap.php")
                {
                        proxy_pass http://count.interface.video.sina.com.cn;
                }

                #按 URL 进行分组，第 2 组：外部提供数据类程序
                if ($request_uri ~ "^\/crossdomain.xml")
                {
                        proxy_pass http://data.interface.video.sina.com.cn;
                }
                if ($request_uri ~ "\/interface\/client\/topVideoClient.php")
                {
                        proxy_pass http://data.interface.video.sina.com.cn;
                }
                if ($request_uri ~ "^\/interface\/common\/")
                {
                        proxy_pass http://data.interface.video.sina.com.cn;
                }
                if ($request_uri ~ "^\/interface\/randplay\/randplay.php")
                {
                        proxy_pass http://data.interface.video.sina.com.cn;
```


www.TopSage.com

大学网 TopSage.com

```
        }
        if ($request_uri ~ "^\/interface\/topic\/suggTopic.php")
        {
              proxy_pass http://data.interface.video.sina.com.cn;
        }
        if ($request_uri ~ "^\/interface\/uploadClient\/")
        {
              proxy_pass http://data.interface.video.sina.com.cn;
        }
        if ($request_uri ~ "^\/interface\/xml\/")
        {
              proxy_pass http://data.interface.video.sina.com.cn;
        }
        if ($request_uri ~ "^\/outinterface\/")
        {
              proxy_pass http://data.interface.video.sina.com.cn;
        }

        #按 URL 进行分组，第 3 组：打日志类程序
        if ($request_uri ~ "^\/interface\/flash\/")
        {
              proxy_pass http://log.interface.video.sina.com.cn;
        }
        if($request_uri~"^\/interface\/playrank\/playrank2008_10.php")
        {
              proxy_pass http://log.interface.video.sina.com.cn;
        }

        #按 URL 进行分组，其他组：功能相关、嵌套页面等未匹配到的 URL
        proxy_pass      http://log.interface.video.sina.com.cn;
}

#定义日志格式
log_format  count  '$remote_addr - $remote_user [$time_local] $request '
                   '"$status" $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';
#打日志
access_log  /data1/logs/interface.log  count;

#允许客户端请求的最大的单个文件字节数
client_max_body_size    10m;

#缓冲区代理缓冲用户端请求的最大字节数 可以理解为先保存到本地再传给用户
client_body_buffer_size  128k;

#跟后端服务器连接的超时时间_发起握手等候响应超时时间
proxy_connect_timeout   600;

#连接成功后_等候后端服务器响应时间_其实已经进入后端的排队等候处理
proxy_read_timeout      600;
```


www.TopSage.com

```
        #后端服务器数据回传时间_就是在规定时间之内后端服务器必须传完所有的数据
        proxy_send_timeout      600;

        #代理请求缓存区_这个缓存区间会保存用户的头信息以供Nginx进行规则处理_一般只要能保存
         下头信息即可
        proxy_buffer_size       8k;

        #同上 告诉Nginx保存单个用的几个Buffer 最大用多大空间
        proxy_buffers           4 32k;

        #如果系统很忙的时候可以申请更大的proxy_buffers 官方推荐*2
        proxy_busy_buffers_size 64k;

        #proxy缓存临时文件的大小
        proxy_temp_file_write_size 64k;
    }
}
```

10.1.2　Nginx 负载均衡在金山逍遥网中的应用

在金山逍遥网（www.xoyo.com）中，前端的负载均衡服务器采用的是 Nginx，两台 Nginx 服务器为一组，承担多种类型的负载均衡服务，两台负载均衡服务器都处于活动状态，各自绑定一个公网虚拟 IP，作为负载均衡服务器，当其中一台服务器发生故障时，另一台服务器接管发生故障服务器的虚拟 IP。这种方式的详细介绍请见第 6 章。

代码 10-2 是 Nginx 负载均衡在金山逍遥网中的配置（nginx.conf）：

代码 10-2

```
user  www www;

worker_processes 8;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 51200;

events
{
 use epoll;
 worker_connections 51200;
}

http
{
```


www.TopSage.com

```
include      mime.types;
default_type  application/octet-stream;

#charset  utf-8;

server_names_hash_bucket_size 128;
client_header_buffer_size 32k;
large_client_header_buffers 4 32k;

sendfile on;
#tcp_nopush    on;

keepalive_timeout 30;

tcp_nodelay on;

fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
fastcgi_buffer_size 64k;
fastcgi_buffers 4 64k;
fastcgi_busy_buffers_size 128k;
fastcgi_temp_file_write_size 128k;

gzip on;
gzip_min_length  1k;
gzip_buffers     4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types     text/plain application/x-javascript text/css application/xml;
gzip_vary on;

limit_zone  anti_attack  $binary_remote_addr  10m;

#允许客户端请求的最大的单个文件字节数
client_max_body_size    300m;

#缓冲区代理缓冲用户端请求的最大字节数 可以理解为先保存到本地再传给用户
client_body_buffer_size  128k;

#跟后端服务器连接的超时时间_发起握手等候响应超时时间
proxy_connect_timeout    600;

#连接成功后_等候后端服务器响应时间_其实已经进入后端的排队之中等候处理
proxy_read_timeout      600;

#后端服务器数据回传时间_就是在规定时间之内后端服务器必须传完所有的数据
proxy_send_timeout      600;

#代理请求缓存区_这个缓存区间会保存用户的头信息以供Nginx进行规则处理_一般只要能保存下头信息即可
proxy_buffer_size       16k;
```


www.TopSage.com

大家网 TopSage.com

```
#同上 告诉 Nginx 保存单个用的几个 Buffer 及最大用多大空间
proxy_buffers          4 32k;

#如果系统很忙的时候可以申请更大的 proxy_buffers 官方推荐*2
proxy_busy_buffers_size 64k;

#proxy 缓存临时文件的大小
proxy_temp_file_write_size 64k;

#缓存
proxy_temp_path /data2/proxy_temp_path;
proxy_cache_path /data2/proxy_cache_path levels=1:2  keys_zone=cache_one:2000m
   inactive=1d max_size=5m;

upstream my_server_pool {
  server   xx.xx.xx.1:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.2:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.3:80 weight=1 max_fails=2 fail_timeout=30s;
}

upstream php_server_pool {
  server   xx.xx.xx.4:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.5:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.6:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.7:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.8:80 weight=1 max_fails=2 fail_timeout=30s;
}

upstream bbs_server_pool {
  ip_hash;
  server   xx.xx.xx.9:80 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.10:80 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.11:80 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.12:80 max_fails=2 fail_timeout=30s;
}

upstream cms_server_pool {
  server   xx.xx.xx.13:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.14:80 weight=1 max_fails=2 fail_timeout=30s;
}

upstream pic_server_pool {
  server   xx.xx.xx.15:80 weight=1 max_fails=2 fail_timeout=30s;
  server   xx.xx.xx.16:80 weight=1 max_fails=2 fail_timeout=30s;
}

upstream xoyohimsg_server_pool {
  server   xx.xx.xx.17:3245;
  server   xx.xx.xx.18:3245 down;
}

#xoyo.com 域名跳转到 www.xoyo.com
```


www.TopSage.com

```
server
{
  listen      80;
  server_name  xoyo.com;

  rewrite ^/(.*) http://www.xoyo.com/ permanent;

  access_log  /data1/logs/xoyo.com_access.log;
}

#用户中心 HTTPS/SSL 加密浏览
server
{
  listen      443;
  server_name  my.xoyo.com;

  ssl on;
  ssl_certificate my.xoyo.com.crt;
  ssl_certificate_key my.xoyo.com.key;

  location /
  {
       proxy_pass http://php_server_pool;
       proxy_set_header Host  my.xoyo.com;
       proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log  /data1/logs/my.xoyo.com_access.log;
}

#图片服务器，不同的路径访问后端不同的服务器
server
{
  listen      80;
  server_name  pic.xoyo.com;

  location /cms/
  {
       proxy_pass http://cms_server_pool;
       proxy_set_header Host  pic.xoyo.com;
             proxy_set_header X-Forwarded-For  $remote_addr;
  }

  location /
  {
       proxy_pass http://pic_server_pool;
       proxy_set_header Host  pic.xoyo.com;
             proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log  /data1/logs/pic.xoyo.com_access.log;
}
```

www.TopSage.com

```
#音频电台文件下载，进行简单防盗链
#limit_zone  media $binary_remote_addr 10m;
server
{
 listen     80;
 server_name media.xoyo.com;

 location /
 {
    proxy_pass http://cms_server_pool;
    proxy_set_header Host  $host;
    proxy_set_header X-Forwarded-For  $remote_addr;

    valid_referers none blocked www.xoyo.com *.xoyo.com www.kingsoft.com
       *.kingsoft.com www.kingsoft.cn *.kingsoft.cn;
    if ($invalid_referer) {
      rewrite ^/ http://www.xoyo.com;
    }
 }

 access_log  /data1/logs/media.xoyo.com_access.log;
}

# “逍遥有聊” WebIM 产品的负载均衡，反向代理两种 HTTP 服务器
server
{
 listen     80;
 server_name hi.xoyo.com;

 #反向代理一款定制开发的高性能消息队列 HTTP 服务器
 location /recmessage.xoyo
 {
      proxy_pass http://xoyohimsg_server_pool;
      proxy_set_header Host  $host;
 }

 location /
 {
      proxy_pass http://php_server_pool;
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
 }

 access_log  /data1/logs/hi.xoyo.com_access.log;
}

#论坛负载均衡，并对图片、Flash、JavaScript、CSS、静态 HTML 进行 Web 缓存
server{
 listen80;
 server_name bbs.xoyo.com *.bbs.xoyo.com bbs.xoyo.kingsoft.com;
```


www.TopSage.com




```
  location /
  {
       proxy_set_header Host  $host;
       proxy_set_header X-Forwarded-For  $remote_addr;
       proxy_pass http://bbs_server_pool;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|js|css|html|shtml)$
  {
       proxy_cache cache_one;
       proxy_cache_valid  200 10m;
       proxy_cache_valid  304 1m;
       proxy_cache_valid  301 302 1h;
       proxy_cache_valid  any 1m;
       proxy_cache_key $host$uri$is_args$args;
       proxy_set_header Host  $host;
       proxy_set_header X-Forwarded-For  $remote_addr;
       proxy_pass http://bbs_server_pool;
  }


  log_format  bbs  '$remote_addr $host $remote_user [$time_local] "$request"'
          '$status $body_bytes_sent "$http_referer" '
          '"$http_user_agent" $http_x_forwarded_for';
  access_log /data1/logs/bbs.xoyo.com_access.log bbs;
}


#论坛附件反向代理，限制下载速度为256KB/秒
server{
  listen      80;
  server_name  att03.bbs.xoyo.com att02.bbs.xoyo.com att01.bbs.xoyo.com;

  location /
  {
       #限制下载速度为256KB/秒
       limit_rate  256k;
       proxy_pass http://xx.xx.xx.19;
       proxy_set_header Host  $host;
       proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log off;
}

#逍遥江湖SNS社区，管理后台定位到一台服务器上，并对图片、Flash、JavaScript、CSS进行Web缓存区
server{
  listen      80;
  server_name hu.xoyo.com *.hu.xoyo.com;

  location /
  {
```

www.TopSage.com

```
      proxy_pass http://php_server_pool;
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
  }

  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|js|css)$
  {
      proxy_cache cache_one;
      proxy_cache_valid  200 10m;
      proxy_cache_valid  304 1m;
      proxy_cache_valid  301 302 1h;
      proxy_cache_valid  any 1m;
      proxy_cache_key $host$uri$is_args$args;
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
      proxy_pass http://php_server_pool;
  }

  location ~ ^/admincp.php
  {
      #管理后台定位到一台服务器上
      proxy_pass http://xx.xx.xx.4;
      proxy_set_header Host  $host;
      proxy_set_header X-Forwarded-For  $remote_addr;
  }

  access_log /data1/logs/hu.xoyo.com_access.log;
 }
}
```

10.2 Nginx+PHP 类网站应用案例

10.2.1 Nginx+PHP 在金山逍遥网 CMS 发布系统中的应用

金山逍遥网 CMS 发布系统采用 PHP 编写，自主开发的 MVC 框架，和其他的 MVC 框架一样，拥有单一请求处理入口/application/cmsmanage/index.php，逻辑清晰，Nginx 的配置也就简单明了：如果文件不存在，则直接 Rewrite（重写）到单一入口文件/application/ cmsmanage/index.php 上。

MVC 是 Model（模型）、View（视图）、Controller（控制器）这三个单词的缩写组合。MVC 是一种普遍的软件敏捷开发模式，在许多领域特别是桌面编程领域早已经得到了广泛的应用，随着众多框架的涌现，MVC 在 PHP 的各个框架中也得到了实现。

单一入口指在一个网站中，所有的动态请求都是指向一个脚本文件的，例如 *http://cms.xoyo.com/application/cmsmanage/index.php*，所有对应用程序的访问都必须通过这个入口。正是单一入口才使得 MVC 模式得以实现，因为当你访问一个 URL 的时候，index.php 会调用基础类库，做

www.TopSage.com

初始化工作，并通过从地址栏的 URL 路径获取参数，加载控制器、视图、模型等内容信息。

代码 10-3 则是金山游戏官方网站——逍遥网的 CMS 发布系统 Nginx 配置：

代码 10-3

```
user  www www;

worker_processes 8;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 51200;

events
{
 use epoll;
 worker_connections 51200;
}

http
{
 include       mime.types;
 default_type  application/octet-stream;

 #charset  utf-8;

 server_names_hash_bucket_size 128;
 client_header_buffer_size 32k;
 large_client_header_buffers 4 32k;
 client_max_body_size 300m;
 client_body_buffer_size 128k;

 sendfile on;
 #tcp_nopush     on;

 keepalive_timeout 65;

 tcp_nodelay on;

 fastcgi_connect_timeout 300;
 fastcgi_send_timeout 300;
 fastcgi_read_timeout 300;
 fastcgi_buffer_size 64k;
 fastcgi_buffers 4 64k;
 fastcgi_busy_buffers_size 128k;
 fastcgi_temp_file_write_size 128k;
```


www.TopSage.com




```
gzip off;
gzip_min_length  1k;
gzip_buffers     4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types       text/plain application/x-javascript text/css application/xml;
gzip_vary on;

#limit_zone  crawler  $binary_remote_addr  10m;

server
{
  listen       80;
  server_name  cms.xoyo.com;
  index index.html index.htm index.php;
  root  /data0/htdocs/cms.xoyo.com;

  location ~ .*\.(sh|bash)?$ { return 403; }

  location / {
      index index.html index.php;
        #如果请求的文件不存在，则重定向到单一入口文件上
      if (!-f $request_filename)
      {
            rewrite ^/(.*)$ /application/cmsmanage/index.php last;
      }
  }

  location ~ .*\.(php|php5)?$
  {
    #fastcgi_pass  unix:/tmp/php-cgi.sock;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
  }

  log_format  cms  '$remote_addr - $remote_user [$time_local] "$request" '
            '$status $body_bytes_sent "$http_referer" '
            '"$http_user_agent" $http_x_forwarded_for';
  access_log  /data1/logs/cms.xoyo.com_access.log  cms;
 }
}
```

10.2.2 Nginx+PHP 在某分类信息网站中的应用

由于历史原因，该网站没有任何开发框架，针对搜索引擎的 URL 优化，兼容旧的 URL 地址，几乎全靠 Rewrite 规则来实现，这也使得该网站 nginx.conf 配置中拥有大量的 Rewrite 规则。虽然这种方法不推荐使用，但是，这些实现各种功能的 Rewrite 规则的应用，具有不小的学习价值。

 www.TopSage.com

由于篇幅有限，本书就不列出配置文件的完整内容了，感兴趣的读者可以访问以下网址查看：

http://blog.s135.com/book/nginx/nginx_rewrite_conf.txt

10.3 Nginx 视频点播类网站应用案例

10.3.1 Nginx 视频点播在金山游戏视频网站中的应用

金山逍遥视频网站为金山软件公司网络游戏的宣传视频，提供 Flash 视频流媒体点播服务。

整个逍遥视频站点由 4 部分组成：

（1）v.xoyo.com（逍遥视频网站页面部分）；

（2）FLV 视频节点部分；

（3）api.v.xoyo.com（逍遥视频 Flash 播放器与 API 接口部分）；

（4）admin.v.xoyo.com（逍遥视频管理后台部分、集成在 CMS 系统中）。

视频发布流程如下：

金山软件公司的视频编辑人员通过 CMS 统一登录入口，登录进入 admin.v.xoyo.com 视频管理后台。在此后台可以上传 FLV 视频、文字描述。视频上传完成后，系统会自动生成视频 ID 编号（即 vid），并将视频 Rsync 推送到各节点服务器。

视频播放流程（以访问 vid=387 的视频为例）如下：

（1）用户访问逍遥视频页面 *http://v.xoyo.com/play/387*，页面内嵌入了 Flash 播放器地址：*http://api.v.xoyo.com/external/player.swf?config=http://api.v.xoyo.com/external/video-387.swf*，其中 *http://api.v.xoyo.com/external/player.swf* 为 Flash 播放器，*http://api.v.xoyo.com/external/video-387.swf* 为经过 Rewrite 重写规则伪静态的视频 API 接口，其真实地址为 *http://api.v.xoyo.com/external/video.php?vid=387*。

（2）视频 API 接口（*http://api.v.xoyo.com/external/video.php?vid=387*）记录视频播放数等信息，并根据用户来源 IP 所属的地域、ISP 网络接入商进行判断，选择一个当前用户下载速度最快的 FLV 视频节点，输出包含 FLV 视频下载地址的 XML 文件，内容如下。该 API 接口还会根据各 FLV 视频节点报告的实时带宽使用情况，在多个 IDC 节点动态调节视频带宽资源。当一个 IDC 节点的使用带宽超过设定值后，新的视频播放请求将自动按照地域、ISP 类型等顺序，分到邻近的带宽尚有空余的其他 IDC，代码如下。

www.TopSage.com

```
<config>
  <icons>false</icons>
  <plugins>xoyo_complete</plugins>
  <complete.site>http://jx3.xoyo.com</complete.site>
  <file>http://219.232.254.222/streams/2009/10/30/387.flv</file>
  <type>nginx</type>
  <image>http://pic.xoyo.com/streams/2009/10/30/387_bg.jpg</image>
</config>
```

（3）Flash 播放器调用、播放 FLV 视频文件。

（4）不同地域、不同 ISP 的用户访问的是离自己最近的 FLV 视频节点上的视频。

（5）能够支持 Flash 播放器进度条拖动到任意位置播放。

图 10-1 为系统架构图。

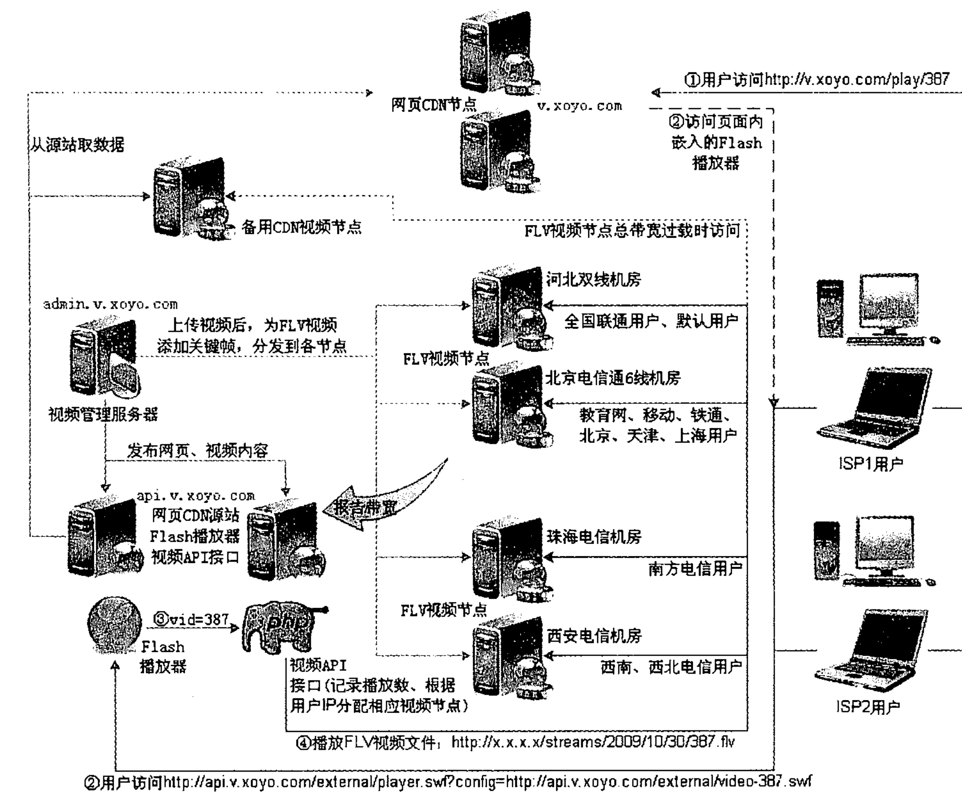


图 10-1 金山逍遥视频点播系统架构图

系统架构图中的视频节点采用的是 Nginx 服务器提供 FLV 视频播放服务，配置过程可以参见第 11 章 11.2.1 节的“采用 Nginx 的 Flv Stream 模块搭建 HTTP 下载方式的 FLV 视频服务器”。

www.TopSage.com

大家网
TopSage.com

第 11 章

Nginx 的非典型应用实例

Nginx 的用途非常多，在第 10 章已经介绍了 Nginx 作为负载均衡服务器、动态应用服务器、静态内容服务器的典型应用案例。而在本章中，主要介绍 Nginx 的一些实用的非典型应用实例。

11.1 用 HTTPS（SSL）构建一个安全的 Nginx Web 服务器

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单来讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

它是一个 URI scheme（抽象标识符体系），句法类同 http:体系，用于安全的 HTTP 数据传输。https:URL 表明它使用了 HTTP，但 HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP 与 TCP 之间）。这个系统的最初研发由网景公司进行，提供身份验证与加密通信方法，现在它被广泛用于万维网上安全敏感的通信，例如交易支付方面。

国内一些大型网站用户中心、邮箱、电子支付等业务，都支持 HTTPS（SSL）加密传输。

www.TopSage.com

11.1.1 自行颁发不受浏览器信任的 SSL 证书

HTTPS 的 SSL 证书可以自行颁发，下面笔者以域名 api.bz 自行颁发一个 SSL 证书为例，介绍 SSL 证书的颁发步骤。

首先，创建一个私钥文件。api.bz.key 和 api.bz_nopass.key 都是私钥文件，不同的是前者须要输入密码，而后者不须要。假设在 Linux 的/etc/rc.local 文件中添加 Nginx 的启动脚本，重启服务器后，Nginx 会自动启动。但是，如果私钥 api.bz.key 用于 Nginx，那么在 Nginx 启动时会提示输入该私钥文件的密码，Nginx 则无法完成自动启动。使用 api.bz_nopass.key 在启动 Nginx 时无须输入密码，代码如下。

```
openssl genrsa -des3 -out api.bz.key 1024
openssl req -new -key api.bz.key -out api.bz.csr
openssl rsa -in api.bz.key -out api.bz_nopass.key
```

以上三条语句的操作步骤如图 11-1 所示。

```
[root@blog gssl]# openssl genrsa -des3 -out api.bz.key 1024
Generating RSA private key, 1024 bit long modulus
.++++++
.......................++++++
e is 65537 (0x10001)
Enter pass phrase for api.bz.key:输入密码
Verifying - Enter pass phrase for api.bz.key:再次输入密码
[root@blog gssl]# openssl req -new -key api.bz.key -out api.bz.csr
Enter pass phrase for api.bz.key:输入刚才设置的密码
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:Beijing
Locality Name (eg, city) [Newbury]:Beijing
Organization Name (eg, company) [My Company Ltd]:API.BZ
Organizational Unit Name (eg, section) []:API.BZ
Common Name (eg, your name or your server's hostname) []:*.api.bz
Email Address []:admin@api.bz

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: 直接回车
An optional company name []:直接回车
[root@blog gssl]# openssl rsa -in api.bz.key -out api.bz_nopass.key
Enter pass phrase for api.bz.key:输入刚才设置的密码
writing RSA key
[root@blog gssl]#
```

图 11-1 自行生成 SSL 私钥

然后，创建一个自签署的 CA 证书：

```
openssl req -new -x509 -days 3650 -key api.bz_nopass.key -out api.bz.crt
```

按照提示，填写国家、省份、城市、公司、部门、域名、邮箱等内容：

```
Country Name (2 letter code) [GB]:CN
State or Province Name (full name) [Berkshire]:Beijing
```


www.TopSage.com

```
Locality Name (eg, city) [Newbury]:Beijing
Organization Name (eg, company) [My Company Ltd]:API.BZ
Organizational Unit Name (eg, section) []:API.BZ
Common Name (eg, your name or your server's hostname) []:*.api.bz
Email Address []:admin@api.bz
```

通过自行颁发私钥文件 api.bz_nopass.key 和 CA 证书 api.bz.crt，就可以搭建安全的 Nginx Web 服务器了。

编译安装 Nginx 服务器时，须要加上 SSL 模块，例如：

```
./configure --user=www --group=www --prefix=/usr/local/webserver/nginx
--with-http_ssl_module
make && make install
```

然后，可以按照代码 11-1 示例内容，在 nginx.conf 配置文件中配置 HTTP（SSL）虚拟主机：

代码 11-1

```
......
server
{
  server_name sms.api.bz;
  listen  443;
  index index.html index.htm index.php;

  root  /data0/htdocs/api.bz;

  ssl on;
  ssl_certificate api.bz.crt;
  ssl_certificate_key api.bz_nopass.key;
  ......
}
......
```

自行颁发的 SSL 证书虽然能够实现加密传输功能，但得不到浏览器的信任，会出现如图 11-2 所示提示。

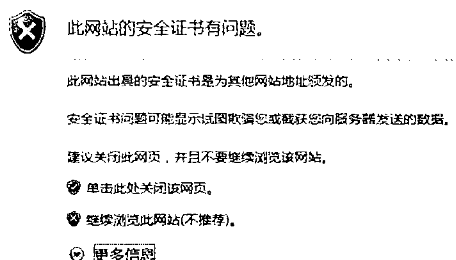


图 11-2 自行颁发的 SSL 证书在 IE7、IE8 浏览器下的提示

要解决浏览器的信任问题，需要由正规的 CA 机构来颁发证书。

www.TopSage.com

大家网 TopSage.com

11.1.2 向 CA 机构申请颁发受浏览器信任的 SSL 证书

能够颁发受浏览器信任的 SSL 证书的 CA 机构有很多家，国外著名的 CA 机构有 VeriSign、GlobalSign、GeoTrust、Thawte、Visa、Microsoft 等，国内的 CA 机构有中国互联网络信息中心（CNNIC）。

向 CA 机构申请 SSL 证书，同样要按照以下步骤操作，先生成自己的私钥文件和 CSR 文件。然后把 CSR 文件（api.bz.csr）提交给 CA 机构，由 CA 机构生成 CRT 证书文件给你。

```
openssl genrsa -des3 -out api.bz.key 1024
openssl req -new -key api.bz.key -out api.bz.csr
openssl rsa -in api.bz.key -out api.bz_nopass.key
```

CA 机构颁发的 SSL 证书，通常单域名和泛域名的价格不一样，所以在使用 openssl 生成 CSR 文件时，须要注意 Common Name 的填写，如果你购买的是单域名 SSL 证书，Common Name 填写的内容就必须和你购买的 SSL 证书域名一致。例如你购买了 sms.api.bz 域名的 SSL 证书，Common Name 就必须填写 sms.api.bz：

```
Common Name (eg, your name or your server's hostname) []:*.api.bz
```

在国内的互联网企业中，对于安全性要求较高的网站，例如招商银行网上银行、中国工商银行网上银行、中国建设银行网上银行、支付宝、百度的百付宝、腾讯的财付通，可选择 VeriSign 作为其 SSL 证书供应商。其他一些网站，例如网易邮箱、金山逍遥网用户中心，采用的是 CNNIC 颁发的 SSL 证书。

当然，上述这些 CA 机构颁发的 SSL 证书是要付费的。目前只有一家 CA 机构，颁发的 SSL 证书是免费的。

跟 VeriSign 一样，StartSSL（网址：*http://www.startssl.com*，公司名：StartCom）也是一家 CA 机构，它的根证书很早之前就被一些具有开源背景的浏览器支持（Firefox 浏览器部分版本、谷歌 Chrome 浏览器、苹果 Safari 浏览器等）。

2009 年 9 月，微软在 Windows 升级补丁中，更新了通过 Windows 根证书认证程序（Windows Root Certificate Program）的厂商清单，并首次将 StartCom 公司列入了该认证清单，这是微软首次将提供免费数字验证技术的厂商加入根证书认证列表中。现在，在 Windows 7 或安装了升级补丁的 Windows Vista 或 Windows XP 操作系统中，系统会完全信任由 StartCom 这类免费数字认证机构认证的数字证书，从而让 StartSSL 也获得 IE 浏览器的支持。

注册成为 StartSSL（*http://www.startssl.com*）用户，并通过邮件验证后，就可以申请免费的可信任的 SSL 证书了。步骤比较复杂，就不详细介绍了，申请向导的主要步骤如下：

（1）在申请向导中选择验证类型为域名验证。StartSSL 将通过发邮件到你的域名 Whois 信

www.TopSage.com

息中的联系人 E-mail 地址，来验证您是否为该域名的拥有者。只有验证通过，才会颁发证书给你，如图 11-3 所示。

Tool Box | Certificates Wizard | Validations Wizard

Select Validation

- Select the type and attribute of validation you'd like to perform.
- Please note that you might need to have instant access to your email account(s) or documents in image format ready.

Type: Domain Name Validation

Continue >>>

This wizard will guide you through the process of validating your domain names. You can always view your validated domain names by opening the "Domain Validations" at the right hand panel.

图 11-3 StartSSL 免费 SSL 证书申请向导（1）

（2）输入你要申请 SSL 证书的域名，如图 11-4 所示。

Tool Box | Certificates Wizard | Validations Wizard

Enter Domain Name

- Enter the domain name you want to have validated.
- You must be the owner of the top-level domain, sub domains are not supported.

http:// api .bz

Continue >>>

图 11-4 StartSSL 免费 SSL 证书申请向导（2）

（3）选择证书使用目的：Web Server SSL，如图 11-5 所示。

Tool Box | Certificates Wizard | Validations Wizard

Select Certificate Purpose

- Make sure you have already validated a domain name or email address before using this tool! Select the **"Validations Wizard"** for this task.
- Depending on your preferences and type of software, you need to have a prepared certificate request (CSR) ready for submission.

Certificate Target: Web Server SSL/TLS Certificate

Continue >>>

SSL/TLS web server certificates are mostly used by web servers to secure communication across a network and to prevent eavesdropping, tampering and message forgery. Additionally server certificates identify the operator of the server at various levels, which allows a relaying party (visitors) to gain verified information about the owners of the web site. This is essential for trading (e-business) on the Internet and provides a trust factor otherwise difficult to obtain.

图 11-5 StartSSL 免费 SSL 证书申请向导（3）

www.TopSage.com

并非所有浏览器都能信任 StartSSL 的证书，如果您的网站比较重要，还是建议您购买 VeriSign、GlobalSign、CNNIC 等商业 CA 机构的 SSL 证书。

11.2 采用 Nginx 搭建 FLV 视频服务器

国内外著名的视频分享网站，例如 YouTube、优酷网、土豆网、新浪播客等，采用的都是 Flash 播放器播放 FLV/MP4 视频文件的技术。

FLV 视频可以采用两种方式发布，一种是普通的 HTTP 下载方式，另一种是基于 Flash Media Server 或 Red5 服务器的 RTMP/RTMPT 流媒体方式。YouTube、优酷网、土豆网、新浪播客等，采用的是 HTTP 下载方式，金山逍遥网的部分游戏视频采用的是 RTMP 流媒体方式。流媒体方式无须下载视频文件到本地，可以实时播放服务器上的 FLV 文件，可以任意拖曳 Flash 播放器播放进度条，用户体验较好，但是比较消耗服务器资源。而 HTTP 下载方式须要下载 FLV 视频文件到本地播放，一旦 FLV 视频下载完成，就不消耗服务器资源和带宽了，虽然也可以实现拖动，但是拖动功能不如流媒体方式强大。

对基于 HTTP 下载方式的 FLV 视频发布，可以通过 Nginx 的 Flv Stream 模块，实现 FLV 视频播放的进度条拖动。对基于 RTMPT 流媒体协议的 FLV 视频发布，Nginx 可以用来实现对后端 Flash Media Server 或 Red5 服务器 RTMPT 的反向代理。

11.2.1 采用 Nginx 的 Flv Stream 模块搭建 HTTP 下载方式的 FLV 视频服务器

Flv Stream 模块是 Nginx 的可选模块，需要在编译安装 Nginx 服务器时，把 Flv Stream 模块加上，例如：

```
./configure --user=www --group=www --prefix=/usr/local/webserver/nginx
--with-http_flv_module
make && make install
```

然后，按照代码 11-2 所示示例方式，在 nginx.conf 配置文件中配置存储 FLV 视频文件的虚拟主机：

代码 11-2

```
......
server
{
  listen       80;
  server_name  flv.domain.com;
  index index.shtml index.html index.htm;
```


www.TopSage.com

大家网
TopSage.com

```
    root  /data0/htdocs/flv_files;

    limit_rate_after 3m;
    limit_rate  512k;

    location ~ \.flv
    {
        flv;
    }

    access_log  off;
}
......
```

重启 Nginx 后，Nginx 配置部分就算 OK 了。limit_rate_after 3m 和 limit_rate 512k 两项指令设置了一开始不限速，在客户端下载 FLV 视频大小超过 3MB 后，开始限制下载速度为 512KB/秒。一开始不限速可以使得刚播放视频时，下载到客户端的视频文件字节数尽量多，用户播放无须长时间等待缓冲。大小超过 3MB 的视频文件一般都比较大，可以限速让用户边观看边下载，在不影响用户体验的情况下，可以节省不少服务器带宽资源。

当用户拖动 Flash 播放器的进度条时，会发起一个类似 *http://flv.domain.com/test.flv?start=12345* 的请求 URL 到 Nginx 服务器，告诉 Nginx 服务器下载 start 参数值附近的关键帧之后的那部分 FLV 文件，从而实现拖动播放。

光配置好 Nginx 服务器不管用，还需要 FLV 文件 MetaData 中含有关键帧信息，才能实现拖动播放。一般从其他视频格式转换成 FLV 视频格式的文件 MetaData 中是不含关键帧信息的，可以在 Linux 下使用开源软件自动为 FLV 文件添加关键帧信息。

Linux 下为 FLV 文件添加关键帧的常用软件有两种。

（1）开源软件 Yamdi：

我们可以按照代码 11-3 所示方式编译安装 Yamdi：

代码 11-3

```
wget
http://sourceforge.net/projects/yamdi/files/yamdi/1.4/yamdi-1.4.tar.gz/download
tar zxvf yamdi-1.4.tar.gz
cd yamdi-1.4/
make
make install
cd ../
```

安装完成 Yamdi 之后，可以按照以下示例为 FLV 文件添加关键帧信息：

```
yamdi -i test.flv -o test.flv.new
mv -f test.flv.new test.flv
```


www.TopSage.com

（2）开源软件 FlvTool2：

在 CentOS 下，可以按照以下方式安装 FlvTool2：

```
    yum install ruby
    wget http://blog.s135.com/soft/linux/flvtool2/flvtool2-1.0.6.tgz
tar zxvf flvtool2-1.0.6.tgz
cd flvtool2-1.0.6/
ruby setup.rb
cd ../
```

安装完成 Yamdi 之后，可以按照以下示例为 FLV 文件添加关键帧信息：

```
flvtool2 -U test.flv
```

采用 FlvTool2 为 FLV 文件添加的关键帧为每两秒钟一个，比 Yamdi 添加的关键帧要多，因此视频拖动播放时要更流畅。但是，FlvTool2 为 FLV 文件添加关键帧所耗费的时间要比 Yamdi 多几倍到几十倍。

为 FLV 视频文件添加完关键帧后，就要找一个支持 HTTP 进度条拖动的 Flash 播放器来播放 FLV 视频了。开源的 JW Player 播放器是个不错的选择，可以通过其官网网站（*http://www.longtailvideo.com/*）下载。

我们可以通过 JW Player，测试播放 Nginx 服务器上的 FLV 视频，如果不出意外，应该可以正常支持播放进度条拖动：

```
http://player.longtailvideo.com/player.swf?type=http&file=
http://flv.domain.com/test.flv
```

11.2.2 采用 Nginx 实现 FMS/Red5 流媒体视频服务器的负载均衡

Adobe 公司官方的 FMS（Flash Media Server）和第三方的开源 Red5 流媒体服务器都支持 RTMP 协议、RTMPT 协议的 FLV 视频流媒体播放。

RTMP 全称 Real Time Messaging Protocol（实时消息传送协议），是 Adobe Systems 公司为 Flash 播放器和服务器之间音频、视频和数据传输开发的私有协议。

RTMPT 协议是一个包装了 RTMP 的 HTTP 协议，它从客户端发送 POST 请求到服务器。由于 HTTP 连接的非持久性本质，为了及时更新状态和数据，RTMPT 需要客户端周期性向服务器轮询，取得服务器或其他客户端产生的通知事件。

通过 Nginx，可以实现多台 FMS/Red5 流媒体服务器 RTMPT 协议流媒体视频播放的负载均衡。nginx.conf 配置示例如代码 11-4 所示：

www.TopSage.com

代码 11-4

```
......
upstream fms_server_pool {
  ip_hash;
  server  192.168.1.3:80 max_fails=2 fail_timeout=30s;
  server  192.168.1.4:80 max_fails=2 fail_timeout=30s;
  server  192.168.1.5:80 max_fails=2 fail_timeout=30s;
}

server
{
  listen     80;
  server_name  flv.domain.com;
  index index.html index.htm;
  root  /data0/htdocs/flv_files;

  proxy_redirect    off;
  proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;

  proxy_connect_timeout    120;
  proxy_send_timeout       120;
  proxy_read_timeout       120;

  proxy_buffer_size        4k;
  proxy_buffers           4 32k;
  proxy_busy_buffers_size   64k;
  proxy_temp_file_write_size 64k;

  location /open/ {
         proxy_pass       http://fms_server_pool;
  }

  location /close/ {
         proxy_pass       http://fms_server_pool;
  }

  location /idle/ {
         proxy_pass       http://fms_server_pool;
  }

  location /send/ {
         proxy_pass       http://fms_server_pool;
  }

  access_log  /data1/logs/rtmpt.log;
 }
......
```

刚才已经介绍过的开源 JW Player 播放器，同样可以用来播放 RTMPT 协议的 FLV 视频，示例如下：

www.TopSage.com

```
http://player.longtailvideo.com/player.swf?streamer=rtmpt://flv.domain.com/play&
file=videos/test.flv
```

播放基于 RTMPT 协议的 FLV 流媒体视频时，可以通过 Nginx 的访问日志/data1/logs/rtmpt.log 来了解整个播放流程。RTMPT 用命令：OPEN、SEND、IDLE、CLOSE 来控制网络连接，当处理 RTMPT 的请求时，所有的 RTMPT 命令使用 POST 方法来处理请求和回应，此外其他命令则可以使用 GET 方法，如代码 11-5 所示。

代码 11-5

```
192.168.1.2 - - [14/Jan/2010:03:12:06 +0800] "POST /open/1 HTTP/1.1" 200 11 "-"
"Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:11 +0800] "POST /idle/1248426880/0 HTTP/1.1" 200
1 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:17 +0800] "POST /send/1248426880/1 HTTP/1.1" 200
3074 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:22 +0800] "POST /send/1248426880/2 HTTP/1.1" 200
289 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:28 +0800] "POST /send/1248426880/3 HTTP/1.1" 200
1 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:34 +0800] "POST /send/1248426880/4 HTTP/1.1" 200
116 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:41 +0800] "POST /send/1248426880/5 HTTP/1.1" 200
534741 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:45 +0800] "POST /send/1248426880/6 HTTP/1.1" 200
745276 "-" "Shockwave Flash"
192.168.1.2 - - [14/Jan/2010:03:12:49 +0800] "POST /send/1248426880/7 HTTP/1.1" 200
784077 "-" "Shockwave Flash"
......
```

通过 Nginx 对 FMS/Red5 流媒体视频服务器的反向代理，既能够实现负载均衡，又能够实现自动故障转移，是一项不错的选择。

11.3 Nginx+PHP+MySQL 在小内存 VPS 服务器上的优化

VPS（全称 Virtual Private Server）是利用最新虚拟化技术在一台物理服务器上创建多个相互隔离的虚拟私有主机。它们以最大化的效率共享硬件、软件许可证及管理资源。对其用户和应用程序来讲，每一个 VPS 平台的运行和管理都与一台独立主机完全相同，因为每个 VPS 均可独立进行重启并拥有自己的 root 访问权限、用户、IP 地址、内存、过程、文件、应用程序、系统函数库及配置文件。

目前提供 VPS 租用服务的提供商越来越多，大有取代虚拟主机之势。VPS 服务器最重要的指标就是内存大小，多个 VPS 服务器可以共享一颗 CPU，但不能共享同一块内存。内存越大，价格越贵。

www.TopSage.com

本节针对 128MB 内存 Linux 操作系统的 VPS，对 Nginx + PHP + MySQL 优化进行介绍。

11.3.1 增加 swap 交换文件

如果您的 VPS 提供商没有为您的 VPS 划分 swap 交换分区，则可以自行创建 swap 交换文件来代替交互分区。

（1）创建并激活 swap 交换文件。

```
cd /var/
dd if=/dev/zero of=swapfile bs=1024 count=262144
/sbin/mkswap swapfile
/sbin/swapon swapfile
```

（2）加到 fstab 文件中让系统引导时自动启动。

```
vi /etc/fstab
```

在末尾增加以下内容：

```
/var/swapfile swap swap defaults 0 0
```

11.3.2 Nginx 的主配置文件（nginx.conf）优化

Nginx 的主配置文件（nginx.conf）优化示例如代码 11-6 所示：

代码 11-6

```
user  www www;


#Nginx 每个进程耗费 10MB~12MB 内存，这里只开启一个 Nginx 进程，节省内存。
worker_processes 1;

error_log  /data1/logs/nginx_error.log  crit;

pid        /usr/local/webserver/nginx/nginx.pid;

#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 51200;

events
{
use epoll;
worker_connections 51200;
}

http
{
```


www.TopSage.com

```
include       mime.types;
default_type  application/octet-stream;

#charset  gb2312;

server_names_hash_bucket_size 128;
client_header_buffer_size 32k;
large_client_header_buffers 4 32k;

sendfile on;
tcp_nopush     on;

keepalive_timeout 60;

tcp_nodelay on;

fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
fastcgi_buffer_size 64k;
fastcgi_buffers 4 64k;
fastcgi_busy_buffers_size 128k;
fastcgi_temp_file_write_size 128k;

#对网页文件、CSS、JS、XML 等启动 gzip 压缩，减少数据传输量，提高访问速度。
gzip on;
gzip_min_length  1k;
gzip_buffers     4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css application/xml;
gzip_vary on;

server
{
  listen       80;
  server_name  www.yourdomain.com;
  index index.html index.htm index.php;
  root  /data0/htdocs/blog;

  location ~ .*\.(php|php5)?$
  {
    #将 Nginx 与 FastCGI 的通信方式由 TCP 改为 Unix Socket。TCP 在高并发访问下比 Unix Socket
     稳定，但 Unix Socket 速度要比 TCP 快。
    fastcgi_pass  unix:/tmp/php-cgi.sock;
    #fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
  }

  #图片更改较少，将它们在浏览器本地缓存 15 天，可以提高下次打开时的加载速度。
  location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
```


www.TopSage.com

大家网 TopSage.com

```
  {
   expires     15d;
  }

  #将 JavaScript、CSS 在浏览器本地缓存 1 天，访问者在看完一篇文章或一页后，再看另一篇文章或另一
   页的内容，无须从服务器再次下载相同的 JavaScript、CSS，提高了页面显示速度。
  location ~ .*\.(js|css)?$
  {
   expires     1d;
  }

  log_format  access  '$remote_addr - $remote_user [$time_local] "$request" '
          '$status $body_bytes_sent "$http_referer" '
          '"$http_user_agent" $http_x_forwarded_for';
  access_log  /data1/logs/access.log  access;
  }
}
```

11.3.3 PHP（FastCGI）的配置优化

（1）php.ini 配置文件中关于 eAccelerator 的优化。只使用 1MB 共享内存，删除所有在最后 3 600 秒内无法存取的脚本缓存，用磁盘辅助进行缓存，如代码 11-7 所示。

代码 11-7

```
[eaccelerator]
zend_extension="/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-
20060613/eaccelerator.so"
eaccelerator.shm_size="1"
eaccelerator.cache_dir="/usr/local/webserver/eaccelerator_cache"
eaccelerator.enable="1"
eaccelerator.optimizer="1"
eaccelerator.check_mtime="1"
eaccelerator.debug="0"
eaccelerator.filter=""
eaccelerator.shm_max="0"
eaccelerator.shm_ttl="3600"
eaccelerator.shm_prune_period="3600"
eaccelerator.shm_only="0"
eaccelerator.compress="1"
eaccelerator.compress_level="9"
eaccelerator.keys = "disk_only"
eaccelerator.sessions = "disk_only"
eaccelerator.content = "disk_only"
```

（2）如果您按照第 5 章安装了 PHP（FastCGI），则可以按照以下方式对 php-fpm.conf 配置进行优化。

修改两项，一是修改以下一行，将启动的 php-cgi 进程数改为 5 个：

www.TopSage.com

```
<value name="max_children">5</value>
```

二是修改以下一行，将 TCP 模式改为 Unix Socket 模式：

```
<value name="listen_address">/tmp/php-cgi.sock</value>
```

11.3.4 MySQL 5.1 配置优化

（1）使用以下参数编译安装的 MySQL 5.1 默认支持 4 种存储引擎：CSV、MRG_MYISAM、MEMORY、MyISAM，不支持 InnoDB 存储引擎。由于内存有限，而 InnoDB 耗费的内存较大，这里推荐使用 MyISAM 存储引擎。

```
./configure --prefix=/usr/local/webserver/mysql/ --enable-assembler
--with-extra-charsets=complex --enable-thread-safe-client --with-big-tables
--with-readline --with-ssl --with-embedded-server --enable-local-infile
make && make install
```

（2）MySQL 5.1 配置文件（my.cnf）优化示例如代码 11-8 所示。

代码 11-8

```
[client]
port    = 3306
socket  = /tmp/mysql.sock

[mysqld]
user    = mysql
port    = 3306
socket  = /tmp/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /usr/local/webserver/mysql/data
open_files_limit    = 600
back_log = 20
max_connections = 100
max_connect_errors = 200
table_cache = 60
external-locking = FALSE
max_allowed_packet = 16M
sort_buffer_size = 128K
join_buffer_size = 128K
thread_cache_size = 10
thread_concurrency = 8
query_cache_size = 0M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default_table_type = MyISAM
thread_stack = 192K
transaction_isolation = READ-UNCOMMITTED
tmp_table_size = 512K
max_heap_table_size = 32M
/usr/local/webserver/mysql/data/slow.log
```


www.TopSage.com

```
/usr/local/webserver/mysql/data/error.log
long_query_time = 1
log_long_format
server-id = 1
#log-bin = /usr/local/mysql/data/binlog
binlog_cache_size = 2M
max_binlog_cache_size = 4M
max_binlog_size = 512M
expire_logs_days = 7
key_buffer_size = 4M
read_buffer_size = 1M
read_rnd_buffer_size = 2M
bulk_insert_buffer_size = 2M
myisam_sort_buffer_size = 4M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

[mysqldump]
quick
max_allowed_packet = 16M
```

11.4 采用 Nginx 搭建正向代理服务器

正向代理就是通常所说的代理，是某台电脑通过一台服务器来上 Internet 网的这种方式，其中这台电脑就叫客户机，这台服务器就叫正向代理服务器，也就是通常所说的代理服务器。一般情况下，客户机必须指定代理服务器（IE 浏览器可在工具→Internet 选项→连接→局域网设置→代理服务器中设置）。

Nginx 正向代理的 nginx.conf 配置文件如代码 11-9 所示：

代码 11-9

```
......
server
{
  listen      8080;
  location / {
     #DNS 解析服务器的 IP 地址
     resolver 8.8.8.8;
     proxy_pass   http://$host$request_uri;
  }

  access_log  /data1/logs/proxy.log;
}
......
```


www.TopSage.com

大学网 TopSage.com

配置完成后，重启 Nginx 使配置生效。然后，你就可以在 IE 浏览器菜单栏中的“工具”→“Internet 选项”→“连接”→“局域网设置”→“代理服务器”中设置代理服务器 IP 地址（假设为 61.1.1.1）和端口，如图 11-6 所示。

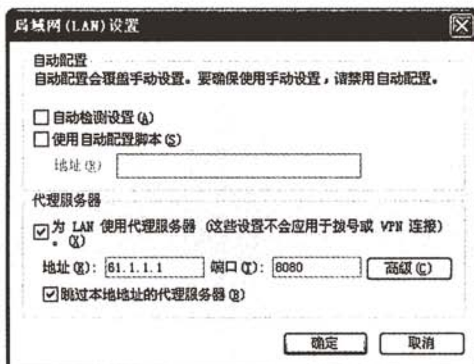


图 11-6 IE 浏览器中的代理服务器设置

然后，您的 IE 浏览器就可以在 Nginx 代理服务器访问 Internet 了。

www.TopSage.com

大家网
TopSage.com

第 12 章

Nginx 的核心模块

在第 4 部分，我们主要介绍 Nginx 的各类模块。Nginx 的模块分为核心模块、标准 HTTP 模块、可选 HTTP 模块、邮件模块、第三方模块和补丁。

其中，Nginx 的核心模块包括主模块和事件模块两部分。

12.1 主模块指令

Nginx 的主模块是实现 Nginx 的基本功能的指令集，它们一般写在 Nginx 的配置文件的最上方。下面，我们来介绍 Nginx 主模块中包含的指令。

12.1.1 daemon 指令

语法：daemon on | off

默认值：on

示例：daemon off;

在生产环境中，请不要使用 daemon 和 master_process 指令。这些选项仅用于开发调试。当然，你也可以在生产环境中设置 daemon off，然后使用进程管理工具启动 Nginx，但是，平滑重启、升级等功能将无法使用。master_process off 绝对不应该用于生产环境。

www.TopSage.com

12.1.2 env 指令

语法：env VAR|VAR=VALUE

默认值：TZ

使用环境：main

此项指令用来定义变量集合，以下场合须更改环境变量，或者添加新的环境变量：

- 在零停机情况下平滑升级 Nginx 时；
- 启用 Nginx 内置 Perl 模块时使用；
- 被 Nginx 进程所使用。

如果没有明确定义 TZ 的值，默认情况下它集成老版本的值，且默认情况下内置的 Perl 模块总是可以使用 TZ 值的。

例如：

```
env  MALLOC_OPTIONS;
env  PERL5LIB=/data/site/modules;
env  OPENSSL_ALLOW_PROXY_CERTS=1;
```

12.1.3 debug_points 指令

语法：debug_points [stop | abort]

默认值：none

示例：debug_points stop;

用于调试，在调试器内设置断点。

12.1.4 error_log 指令

语法：error_log file [debug | info | notice | warn | error | crit]

默认值：${prefix}/logs/error.log

示例：debug_points stop;

file 参数用来指定记录 Nginx 及 FastCGI 错误日志的文件路径。错误日志记录了服务器运行期间遇到的各种错误，以及一些普通的诊断信息，我们可以设置日志文件记录信息级别的高低，

www.TopSage.com

控制日志文件记录信息的数量和类型。通常情况下，Nginx 分为 6 个错误级别，其中 debug 级别最低，记录的错误日志数量最多，范围最广；而 crit 级别最高，只记录非常严重的错误，一般在生产环境中使用。

日志中默认的错误级别：

- main 部分：error
- HTTP 部分：crit
- server 部分：crit

Nginx 支持将不同虚拟主机的日志存储在不同的位置，这是个很有特色的功能。在 lighttpd 中，它们一直拒绝提供类似的功能。代码 12-1 为针对不同虚拟主机提供不同日志的示例：

代码 12-1

```
error_log logs/main_error.log;

events {
 worker_connections 51200;
}

http {
 error_log logs/http_error.log error;
 server {
  server_name one.org;
  access_log logs/one.access;
  error_log logs/one.error error;
 }

 server {
  server_name two.org;
  access_log logs/two.access;
  error_log logs/two.error error;
 }
}
```

如果你在编译 Nginx 的时候，使用了--with-debug 指令，则还可以使用：

```
error_log LOGFILE [debug_core | debug_alloc | debug_mutex | debug_event | debug_http
| debug_imap];
```

注意：error_log off 无法禁用日志，这种写法将会创建一个名为 off 的日志文件。如果要禁用日志，请用下面的写法：

```
error_log /dev/null crit;
```


www.TopSage.com

大家网 TopSage.com

12.1.5 log_not_found 指令

语法：log_not_found on | off

默认值：on

使用环境：debug_points stop;

启用或禁用 404 错误日志，这个指令可以用来禁止 Nginx 记录找不到 robots.txt 和 favicon.ico 这类文件的错误信息。

示例：

```
location = /robots.txt {
  log_not_found off;
}
```

12.1.6 include 指令

语法：linclude file | *

默认值：无

使用此指令，可以包含任何你想要包含的配置文件。从 0.4.4 开始，include 指令开始支持文件名匹配，例如：

```
include vhosts/*.conf;
```

注意：直到0.6.7版本为止，include 文件的路径是相对于 configure 时由--prefix=<PATH>指令指定的路径而言的，默认情况下是/usr/local/nginx。如果在编译 compiledNginx 时你没有指定这个值，请使用绝对路径。

从 0.6.7 开始，include 文件的路径实现归于 Nginx 配置文件 nginx.conf 的所在目录，不再是 Nginx 编译时指定的路径。这个改进大大增加了 include 的灵活性。

12.1.7 lock_file 指令

语法：lock_file file

默认值：compile-time option

示例：lock_file /var/log/lock_file;

如果 Nginx 是由 gcc、Intel C++或 SunPro C++ 在 i386、amd64 平台上编译的，Nginx 可采用异步互斥进行访问控制。

www.TopSage.com

12.1.8 master_process 指令

语法：master_process on | off

默认值：on

示例：master_process off;

生产环境中，请不要使用 daemon 和 master_process 指令，这些选项主要用于开发调试。

12.1.9 pid 指令

语法：pid file

默认值：compile-time option

示例：pid /var/log/nginx.pid;

pid 文件内记录着当前 Nginx 主进程的 ID 号，可以通过 kill 命令发送信号给该 ID 号，例如重新加载 Nginx 配置文件完成平滑重启：kill -HUP `cat /var/log/nginx.pid`

12.1.10 ssl_engine 指令

语法：ssl_engine engine

默认值：系统默认依赖的引擎

此指令可以设置首选的 SSL 引擎。你可以通过命令行工具 openssl engine -t 找出系统目前支持的 SSL 引擎，例如：

```
$ openssl engine -t
(cryptodev) BSD cryptodev engine
  [ available ]
(dynamic) Dynamic engine loading support
```

12.1.11 timer_resolution 指令

语法：timer_resolution t

默认值：none

示例：timer_resolution   100ms;

该指令可以减少 gettimeofday()函数获取当前时间的系统调用次数。在默认情况下，

www.TopSage.com

gettimeofday()函数会在每一次 kevent()、epoll、/dev/poll、select()、poll()返回时被调用。如果你需要在日志中记录毫秒级的准确时间，或者毫秒级的准确反向代理响应时间，你要使用此指令。

12.1.12 try_files 指令

语法：try_files path1 [path2] uri

默认值：none

备注：从 Nginx 0.7.27 版本开始提供此指令。

该指令可以按照参数顺序检查文件是否存在，以及返回第一个被找到的文件名。以“/”结尾的表示一个目录。如果文件没有找到，将被内部重定向到最后一个参数上。最后的参数是一个后备的 URI（统一资源标识符），它必须存在，否则会报内部错误。

代码 12-2 为在反向代理中使用 try_files 指令：

代码 12-2

```
location / {
  try_files /system/maintenance.html
  $uri $uri/index.html $uri.html @mongrel;
}

location @mongrel {
  proxy_pass http://mongrel;
}
```

代码 12-3 为在 FastCGI 中使用 try_files 指令：

代码 12-3

```
location / {
  try_files $uri $uri/ @drupal;
}

location ~ \.php$ {
  try_files $uri @drupal;
  fastcgi_pass 127.0.0.1:8888;
  fastcgi_param  SCRIPT_FILENAME  /path/to$fastcgi_script_name;
  # other fastcgi_param
}

location @drupal {
  fastcgi_pass 127.0.0.1:8888;
  fastcgi_param  SCRIPT_FILENAME  /path/to/index.php;
  fastcgi_param  QUERY_STRING     q=$request_uri;
  # other fastcgi_param
}
```


www.TopSage.com

大家网
TopSage.com

在以上两个例子中，指令 try_files：

```
location / {
  try_files $uri $uri/ @drupal;
}
```

类似于以下的指令：

```
location / {
  error_page    404 = @drupal;
  log_not_found  off;
}
```

和

```
location ~ \.php$ {
  try_files $uri @drupal;

  fastcgi_pass 127.0.0.1:8888;
  fastcgi_param  SCRIPT_FILENAME  /path/to$fastcgi_script_name;
  # other fastcgi_param
}
```

try_files 指令还可以在请求提交给 PHP 等 FastCGI 服务器之前，检查 PHP 文件是否存在。

在 Wordpress 博客系统和 Joomla 内容发布系统中使用的示例如代码 12-4 所示：

代码 12-4

```
location / {
  try_files $uri $uri/ @wordpress;
}

location ~ \.php$ {
  try_files $uri @wordpress;

  fastcgi_pass 127.0.0.1:8888;
  fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;
  # other fastcgi_param
}

location @wordpress {
    fastcgi_pass 127.0.0.1:8888;
    fastcgi_param SCRIPT_FILENAME /path/to/index.php;
    # other fastcgi_param
}
```

12.1.13 user 指令

语法：user user [group]

www.TopSage.com

默认值：nobody nobody

示例：user www users;

该指令用于指定运行 Nginx Worker 进程的用户和组，默认的用户名和组名都是 nobody。如果组名没有指定，Nginx 则默认组名与用户名相同。

12.1.14 worker_cpu_affinity 指令

语法：user user [group]

默认值：nobody nobody

备注：只能在 Linux 环境中使用。

使用此指令，你可以为每个 Nginx Worker 进程绑定指定的一颗 CPU（双核 CPU 算做两颗），Nginx 采用 sched_setaffinity()函数绑定 CPU。

示例如下：

```
worker_processes     4;
worker_cpu_affinity 0001 0010 0100 1000;
```

分别给每个 worker 进程绑定一个 CPU：

```
worker_processes     2;
worker_cpu_affinity 0101 1010;
```

在上例中，第一个 Nginx Worker 进程绑定了 CPU0/CPU2，第二个 Nginx Worker 进程绑定了 CPU1/CPU3。这种配置适用于超线程技术。

12.1.15 worker_priority 指令

语法：worker_priority [-] number

默认值：on

使用该选项可以给所有的 worker 进程分配优先值。

12.1.16 worker_processes 指令

语法：worker_processes number

默认值：1

www.TopSage.com

示例：worker_processes 5;

Nginx 可以使用多个 worker 进程，原因如下：

（1）使用 SMP 对称多处理方式，充分发挥多核 CPU 的优势。SMP 的全称是“对称多处理”（Symmetrical Multi-Processing）技术，是指在一个计算机上汇集了一组处理器（多 CPU），各 CPU 之间共享内存子系统及总线结构。它是相对非对称多处理技术而言的、应用十分广泛的并行技术。在这种架构中，一台电脑不再由单个 CPU 组成，而同时由多个处理器运行操作系统的单一复本，并共享内存和一台计算机的其他资源。虽然同时使用多个 CPU，但是从管理的角度来看，它们的表现就像一台单机一样。系统将任务队列对称地分布于多个 CPU 之上，从而极大地提高了整个系统的数据处理能力。所有的处理器都可以平等地访问内存、I/O 和外部中断。在对称多处理系统中，系统资源被系统中所有 CPU 共享，工作负载能够均匀地分配到所有可用处理器之上。

（2）能够减少进程在磁盘 I/O 阻塞上的延迟时间。

（3）当使用 select()/poll()模型时，能够限制每个进程的连接数。

通过 worker_processes 和 worker_connections 两个指令可以计算出最大客户端连接数，计算公式如下：

```
max_clients = worker_processes * worker_connections
```

12.1.17　worker_rlimit_core 指令

语法：worker_processes number

该指令用于指定每个 Nginx 进程的最大 core 文件大小。

12.1.18　worker_rlimit_nofile 指令

语法：worker_rlimit_nofile limit

示例：worker_rlimit_nofile 65535;

该指令用于指定 Nginx 进程可以打开的最大文件描述符数量。

12.1.19　worker_rlimit_sigpending 指令

语法：worker_rlimit_sigpending limit

示例：worker_rlimit_sigpending 32768;

www.TopSage.com

大家网 TopSage.com

（Linux 2.6.8 以上内核支持）该指令指定调用进程的真正用户 ID 的排队数量。

12.1.20 working_directory 指令

语法：working_directory path

默认值：--prefix

该指令用于指定 Nginx 的工作目录，path 参数只能使用绝对路径。该指令的默认值为 Nginx 使用 configure 文件编译安装时，参数--prefix==PATH 指定的路径。

12.2 主模块变量

在主模块指令中，可以从以下变量获取相关信息：

```
$nginx_version
```

当前运行的 Nginx 版本号。

```
$pid
```

进程 ID 号。

```
$realpath_root
```

Root 目录绝对路径。

12.3 事件模块指令

Nginx 的事件模块是控制 Nginx 处理访问连接的指令集，跟主模块指令一样，事件模块的指令也写在 Nginx 的配置文件的最上方区域。下面，我们来介绍 Nginx 事件模块中包含的指令。

12.3.1 accept_mutex 指令

语法：accept_mutex [on | off]

默认值：on

Nginx 使用连接互斥锁进行顺序的 accept()系统调用。

www.TopSage.com

12.3.2 accept_mutex_delay 指令

语法：accept_mutex_delay Nms;

默认值：500ms

如果一个工作进程没有互斥锁，它将在最少 N 毫秒延迟之后再次尝试获取互斥锁。默认的延迟时间为 500 毫秒。

12.3.3 debug_connection 指令

语法：debug_connection [ip | CIDR]

默认值：none

从 Nginx 0.3.54 版本开始支持 CIDR 地址格式。这个选项用于记录 IP/网络的用户端侦错日志。示例如下：

```
error_log /var/log/nginx/errors;
events {
  debug_connection  192.168.1.1;
}
```

12.3.4 use 指令

语法：use [kqueue | rtsig | epoll | /dev/poll | select | poll | eventport]

如果你在使用./configure 脚步编译安装 Nginx 时，指定了一个以上的事件模型，则要告诉 Nginx 使用哪种事件模型。如果您的操作系统是 FreeBSD 4.1 以上版本，推荐使用 Kqueue 模型，如果您的操作系统是 Linux 2.6 以上内核版本，推荐使用 epoll 模型，如果您的操作系统是 Solaris 10，推荐使用 eventport 模型。Kqueue、epoll、eventport 分别属于 FreeBSD、Linux、Solaris 操作系统中高效的时间模型。

12.3.5 worker_connections 指令

语法：worker_connections number

该指令用于设置每个工作进程能够处理的连接数。通过 worker_connections 和 worker_proceses（Nginx 工作进程数）可以计算出 Nginx 服务器能够处理的最大连接数 max_clients，计算公式如下：

```
max_clients = worker_processes * worker_connections
```

www.TopSage.com

假设你开启了 8 个 Nginx 进程（worker_proceses），设置的 worker_connections 连接数为 32 768，那么最大连接数 max_clients 为 8 * 32768 = 262 144。这只是理论值，实际情况下，受操作系统、文件描述符与端口数的限制，最大连接数是达不到 262 144 的。

在反向代理情况下，最大连接数变成了：

```
max_clients = worker_processes * worker_connections/4
```

因为，Internet Explorer 浏览器通常默认打开两个与服务器的连接，而 Nginx 使用同一个文件描述符池中的描述符来连接后端。

www.TopSage.com

大家网
TopSage.com

第 13 章 Nginx 的标准 HTTP 模块

本章介绍的 HTTP 模块会在编译 Nginx 时自动编译进来，除非使用 configure 命令禁止编译这些模块。

在本章的指令介绍中，指令的“使用环境”是该指令可以在 Nginx 配置文件中使用的位置，例如使用环境为“http, server, location”，表示该指令可以在以下位置使用：

http { ...... }大括号内；server { ...... }大括号内；location { ...... }大括号内。

13.1 HTTP 的核心模块

13.1.1 alias 指令

语法：alias file-path|directory-path;

默认值：no

使用环境：location

该指令用于在 URL 和文件系统路径之间实现映射。它与 root 指令类似，但是网页文件的 root 根目录不会改变，改变的只是请求 URL 的文件系统路径。

示例如下：

www.TopSage.com

```
location /i/ { alias /spool/w3/images/; }
```

在示例中，访问 URL 地址“/i/top.gif”会返回文件“/spool/w3/images/top.gif”。

在被替换的路径中，可以使用变量。在含有正则表达式的 location 中，不能使用 alias 指令，如果你想使用，可以使用 rewrite 和 root 指令的组合来实现。

13.1.2 client_body_in_file_only 指令

语法：client_body_in_file_only on|off

默认值：off

使用环境：http, server, location

该指令允许将一个客户端的请求内容记录到一个文件中，该文件在请求完成后不会被删除。在内置 Perl 中，该指令可以用于调试$r->request_body_file 方法。

13.1.3 client_body_in_single_buffer 指令

语法：client_body_in_single_buffer

默认值：off

使用环境：http, server, location

该指令（0.7.58 以上版本支持）指定是否保持整个内容在一个单一的客户端请求缓冲区中。该指令在使用变量$request_body 时被推荐使用。

13.1.4 client_body_buffer_size 指令

语法：client_body_buffer_size the_size

默认值：8k/16k

使用环境：http, server, location

示例：client_body_buffer_size 128k;

该指令指定客户端请求内容的缓冲区大小。如果客户端请求内容大于缓冲区，整个请求内容或部分请求内容将被写入临时文件。缓冲区默认大小相当于网页大小的两倍，为 8k 或 16k。

www.TopSage.com

13.1.5 client_body_temp_path 指令

语法：client_body_temp_path dir-path [level1 [level2 [level3]

默认值：client_body_temp

使用环境：http, server, location

该指令用于指定存放请求内容临时文件的目录。缓存目录最多支持 3 层子目录。

示例：

client_body_temp_path /spool/nginx/client_temp 1 2;

以上示例的目录结构类似：

/spool/nginx/client_temp/7/45/00000123457

13.1.6 client_body_timeout 指令

语法：client_body_timeout time

默认值：60

使用环境：http, server, location

该指令用于设置读取客户端请求内容的超时时间。如果超过该指令设置的时间，Nginx 将返回"Request time out" 错误信息（HTTP 状态码为 408）。

13.1.7 client_header_buffer_size 指令

语法：client_header_buffer_size size

默认值：1k

使用环境：http, server

该指令用于设置客户端请求的 Header 头缓冲区大小。对绝大多数请求来说，1KB 大小的 Header 头缓冲区已经足够，但是对于 Cookie 内容较大的请求来说，可能不够用，你可以加大该值。

13.1.8 client_header_timeout 指令

语法：client_header_timeout time

www.TopSage.com



默认值：60

使用环境：http, server

该指令用于设置读取客户端请求 Header 头信息的超时时间。如果超过该指令设置的时间，Nginx 将返回“Request time out” 错误信息（HTTP 状态码为 408）。

13.1.9 client_max_body_size 指令

语法：client_max_body_size size

默认值：client_max_body_size 1m

使用环境：http, server, location

示例：client_max_body_size 300m;

该指令用于设置允许接受的客户端请求内容的最大值，即客户端请求 Header 头信息中设置的 Content-Length 的最大值。如果超过该指令设置的最大值，Nginx 将返回“Request Entity Too Large”错误信息（HTTP 状态码为 413）。当默认值为 1MB 时，如果 Nginx 服务器提供上传 1MB 以上的大文件等操作，则要加大该值。

13.1.10 default_type 指令

语法：default_type MIME-type

默认值：default_type text/plain

使用环境：http, server, location

MIME-type 是用来告诉浏览器请求的文件媒体类型的，例如 MIME-type 名 text/plain 表示该文件为文本文件，text/html 表示该文件为 HTML 网页文件。如果 Nginx 无法识别该文件属于何种 MIME-type 类型，则将该文件标记为 default_type 指令设置的 MIME-type。

例如：

```
location = /proxy.pac { default_type application/x-ns-proxy-autoconfig; }
location = /wpad.dat { rewrite . /proxy.pac; default_type
application/x-ns-proxy-autoconfig; }
```

13.1.11 directio 指令

语法：directio [size|off]

www.TopSage.com

默认值：directio off

使用环境：http, server, location

该指令用于设置一个文件大小，当读取的文件超过该大小时，将使用 O_DIRECT 标签（FreeBSD, Linux）、F_NOCACHE 标签（Mac OS X）或 directio() 函数（Solaris）来读取该文件。打开文件时如果用 O_DIRECT，系统就不会使用 buffer 缓冲区，而直接通过 DMA 读取文件。DMA 的英文拼写是“Direct Memory Access”，汉语的意思就是直接内存访问，是一种不经过 CPU 而直接从内存存取数据的数据交换模式。

directio 指令将使 sendfile 功能失效。对于大文件来说，使用该指令是有益的。在以下的示例中，设置了对 4MB 以上的文件使用 O_DIRECT、F_NOCACHE 标签或 directio() 函数读取：

```
directio 4m;
```

13.1.12 error_page 指令

语法：error_page code [code...] [= |=answer-code] uri

默认值：no

使用环境：http, server, location, if in location

该指令用于设置如果出现指定的 HTTP 错误状态码，则返回给客户端显示的对应 URI 地址。

示例如下：

```
error_page 404 /404.html;
```

如果遇到 404 错误状态码（客户端请求的页面不存在），则显示指定的/404.html 文件内容给客户端。注意/404.html 文件大小不能超过 512 字节，否则 Internet Explorer 浏览器会默认为其错误页面，而不是指定的/404.html 页面。

```
error_page 502 503 504 /50x.html;
```

如果遇到 502、503、504 错误状态码（服务器错误），则显示指定的/50x.html 文件内容给客户端。

```
error_page 403 http://example.com/forbidden.html;
```

如果遇到 403 错误状态码（服务器已经理解请求，但是拒绝执行它），则跳转到 URL 地址 *http://example.com/forbidden.html*。

此外，还可以更改 HTTP 的错误响应码为别的响应码，以便某些客户端浏览器能够支持，例如：

```
error_page 404 =200 /empty.gif;
```

如果遇到 404 错误状态码（客户端请求的页面不存在），将状态码改为 200，并显示指定的

www.TopSage.com

/empty.gif 文件内容给客户端。

如果响应错误代码的页面是 PHP 等 FastCGI 程序，则最好在 error_page 中加上=号。

```
error_page 404 = /404.php;
```

如果你想返回原样的错误状态码，则去掉 error_page 指令中的=号。

```
error_page 404 /404.php;
```

13.1.13 if_modified_since 指令

语法：if_modified_since [off|exact|before]

默认值：if_modified_since exact

使用环境：http, server, location

if-modified-since 是由客户端浏览器往 Nginx 服务器发送的头信息。当再次请求本地存在的缓存页面时，客户端浏览器会通过 if-modified-since 头信息将先前 Nginx 服务器端发过来的 Last-Modified 最后修改时间戳发送回去，这是为了让 Nginx 服务器端进行验证，通过这个时间戳判断客户端的页面是否是最新的。如果不是最新的，则返回新的内容；如果是最新的，则返回 304 状态码告诉客户端浏览器其本地缓存的页面是最新的，于是浏览器就可以直接从本地加载页面，这样在网络上传输的数据就会大大减少，从而可加快页面的显示速度，同时也减轻服务器的负担。

if_modified_since 指令（Nginx 0.7.24 以上版本支持）用于设置如何去比较客户端请求 Header 头信息中的“if-modified-since”文件的修改时间。

- off——不检查“if-modified-since”请求头；
- exact——时间完全符合；
- before——文件修改时间应该早于“if-modified-since”请求头中的时间。

13.1.14 index 指令

语法：index file [file...]

默认值：index index.html

使用环境：http, server, location

该指令用于设置 Nginx 的默认首页文件。

例 1：在文件名中可以使用变量，并且最后的默认首页文件可以使用绝对路径（即/index.html）：

www.TopSage.com

大学网 TopSage.com

```
index index.$geo.html index.0.html /index.html;
```

例 2：

```
index index.shtml index.html index.htm index.php default.html;
```

如果某个目录下没有指定的默认首页文件，则可以开启以下指令，显示当前目录的文件列表：

```
autoindex on;
```

13.1.15 internal 指令

语法：internal

默认值：no

使用环境：location

该指令用于设置某个 location 路径只能在 Nginx 内部使用，外部无法访问。

例 1：

```
error_page 404 /404.html;
location /404.html { internal; }
```

例 2：

```
location / {
root /var/www/html;
error_page 404 @40x;
}
location @40x {
root /var/www/errors/40x.html;
}
```

13.1.16 keepalive_timeout 指令

语法：keepalive_timeout [time]

默认值：keepalive_timeout 75

使用环境：http, server, location

keep-alive 功能可使客户端到服务器端的连接持续有效，当出现对服务器的后继请求时，keep-alive 功能可避免建立或重新建立连接。市场上的大部分 Web 服务器，包括 IIS、Apache、Nginx、Lighttpd，都支持 HTTP keep-alive。对于提供静态内容的网站来说，这个功能通常很有用。但是，对于负担较重的网站来说，这里存在另外一个问题：虽然为客户保留打开的连接有

www.TopSage.com

一定的好处，但它同样影响了性能，因为在暂停期间，本来可以释放的资源仍旧被占用。当 Web 服务器和应用服务器在同一台机器上运行时，keep-alive 功能对资源利用的影响尤其突出。此功能为 HTTP 1.1 预设的功能，HTTP 1.0 加上 keep-alive header 也可以提供 HTTP 的持续作用功能。

该指令用于设置 keep-alive 连接超时时间，使用该指令第二个参数设置的时间（秒）将显示在输出的 Header 头 keep-alive: timeout=time 中。不过，一些浏览器不支持 keep-alive。

另外，在 IE 浏览器中，浏览器自身限制 keepalive 时间为 60 秒。无论是客户端浏览器（Internet Explorer）还是 Web 服务器都具有较低的 keepalive 值，这些都将是限制因素。例如，如果客户端的超时值是两分钟，而 Web 服务器的超时值是一分钟，则最大超时值是一分钟。客户端或服务器都可以是限制因素。

13.1.17 keepalive_requests 指令

语法：keepalive_requests n

默认值：keepalive_requests 100

使用环境：http, server, location

设置一个 keep-alive 连接使用的次数。一次请求结束后，如果该连接使用的次数没有超过 keepalive_requests 指令设置的请求次数，则服务器并不立即主动断开连接，而是直到达到 keepalive_timeout 指令设置的时间，才关闭连接。

13.1.18 large_client_header_buffers 指令

语法：large_client_header_buffers number size

默认值：large_client_header_buffers 4 4k/8k

使用环境：http, server

该指令用于设置客户端请求的 Header 头缓冲区大小，默认值为 4KB。客户端请求行不能超过 large_client_header_buffers 指令设置的值，客户端请求的 Header 头信息不能大于 large_client_header_buffers 指令设置的缓冲区大小，否则会报“Request URI too large”（414）或“Bad request”（400）错误。如果客户端的 Cookie 信息较大，则须增加缓冲区大小，示例如下：

```
client_header_buffer_size 128k;
large_client_header_buffers 4 128k;
```


www.TopSage.com

13.1.19 limit_except 指令

语法：limit_except methods {...}

默认值：no

使用环境：location

该指令用于限制 HTTP 方法访问 location 中的内容，示例如下：

```
limit_except GET {
   allow 192.168.1.0/32;
   deny all;
}
```

13.1.20 limit_rate 指令

语法：limit_rate speed

默认值：no

使用环境：http, server, location, if in location

该指令主要用来限速，限速单位是“字节数/秒”，一般在提供 HTTP 下载等应用中会用到该指令。限速只对一个连接起效，如果客户端开启两个连接下载，下载的速度将是限速值的两倍。

例 1：限制每个连接的下载速度为 100KB/秒：

```
limit_rate 100k;
```

例 2：在特定条件下开启限速功能：

```
server {
   if ($slow)
   {
      set $limit_rate 4k;
   }
}
```

13.1.21 limit_rate_after 指令

语法：limit_rate_after time

默认值：limit_rate_after 1m

使用环境：http, server, location, if in location

www.TopSage.com

大家网 TopSage.com

该指令可以设置一个字节数（例如 1MB），下载的字节数大于该值后，limit_rate 指令的限速功能将起效。对于 MP3 在线播放、HTTP 方式的 Flash FLV 视频点播等应用，使用该指令将会起到不错的效果。

例：下载的文件字节数超过 1MB 后，限速为 100KB/秒。

```
limit_rate_after 1m;
limit_rate 100k;
```

13.1.22 listen 指令

语法：listen address:port [default [backlog=num | rcvbuf=size | sndbuf=size | accept_filter=filter | deferred | bind | ssl]

默认值：listen 80

使用环境：server

该指令用于设置虚拟主机监听的服务器地址和端口号。你可以同时设置服务器地址和端口号，也可以只指定一个 IP 地址，或者一个端口号，或者一个服务器名。如果 listen 指令只设置一个服务器名或 IP 地址，那么它的默认端口号为 80。

示例如下：

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;
```

监听 IPV6 地址示例如下：

```
listen [::]:8000;
listen [fe80::1];
```

示例：通过以下配置，可以设置一个虚拟主机同时支持 HTTP 和 HTTPS：

```
listen 80;
listen 443 default ssl;
```

13.1.23 location 指令

语法：location [=|~|~*|^~] /uri/ { ... }

默认值：no

使用环境：server

www.TopSage.com

该指令允许对不同的 URI 进行不同的配置，既可以使用字符串，也可以使用正则表达式。使用正则表达式，须使用以下前缀：

（1）~*，表示不区分大小写的匹配。

（2）~，表示区分大小写的匹配。

在匹配过程中，Nginx 将首先匹配字符串，然后再匹配正则表达式。匹配到第一个正则表达式后，会停止搜索。如果匹配到正则表达式，则使用正则表达式的搜索结果，如果没有匹配到正则表达式，则使用字符串的搜索结果。

可以使用前缀“^~”来禁止匹配到字符串后，再去检查正则表达式。匹配到 URI 后，将停止查询。

使用前缀“=”可以进行精确的 URI 匹配，如果找到匹配的 URI，则停止查询。例如“location =/”，只能匹配到“/”，而“/test.html”则不能被匹配。

正则表达式的匹配，按照它们在配置文件中的顺序进行，写在前面的优先。

示例如代码 13-1 所示：

代码 13-1

```
location  = / {
  # 仅仅匹配/
  [ configuration A ]
}
location  / {
  # 匹配任何以/开头的查询，但是正则表达式及较长的字符串（例如/admin/）将被优先匹配。
  [ configuration B ]
}
location ^~ /images/ {
  # 匹配任何以/images/开头的字符串，并且停止搜索，所以正则表达式将不会被检查。
  [ configuration C ]
}
location ~* \.(gif|jpg|jpeg)$ {
  # 匹配以.gif、.jpg、.jpeg 结尾的任何请求。但是，/images/内的请求将使用 Configuration C 的
    配置。
  [ configuration D ]
}
```

请求处理匹配结果示例：

- / -> configuration A；
- /documents/document.html -> configuration B；
- /images/1.gif -> configuration C；

www.TopSage.com

- /documents/1.jpg -> configuration D。

你可以采用不同的顺序定义这 4 个配置，但是，匹配结果是一样的。

另外，前缀“@”是一个命名标记，这种 location 不会用于正常的请求，它们通常只用于处理内部的重定向（例如：error_page、try_files），示例代码如 13-2 所示：

代码 13-2

```
location ~ \.php$ {
   root   /home/www/htdocs/;
   index  index.php index.html index.htm;
   error_page 404 502 504 @fetch;
}

location @fetch {
   internal;
   proxy_pass http://backend;
   break;
}
```

13.1.24 log_not_found 指令

语法：log_not_found [on|off]

默认值：log_not_found on

使用环境：http, server, location

该指令用来启用或禁用 404 错误日志，这个指令可以用来禁止 Nginx 记录找不到 robots.txt 和 favicon.ico 这类文件的错误信息。

13.1.25 log_subrequest 指令

语法：log_subrequest [on|off]

默认值：log_subrequest off

使用环境：http, server, location

该指令用来启动或禁止在 access_log 中记录类似 rewrite rules、SSI requests 等子请求。

13.1.26 msie_padding 指令

语法：msie_padding [on|off]

www.TopSage.com

大学网 TopSage.com

默认值：msie_padding on

使用环境：http, server, location

此指令关闭或开启 MSIE 浏览器的 msie_padding 特性，若启用此选项，Nginx 会为 response 头部填满至 512 字节，这样就阻止了相关浏览器激活友好错误页面，因此不会隐藏更多的错误信息。

13.1.27 msie_refresh 指令

语法：msie_refresh [on|off]

默认值：msie_refresh off

使用环境：http, server, location

此指令允许或禁止为 MSIE 指派一个 refresh 而不是重定向。

13.1.28 open_file_cache 指令

语法：open_file_cache max = N [inactive = time] | off

默认值：open_file_cache off

使用环境：http, server, location

该指令用于设置打开文件的缓存。以下信息可以被缓存：

- 打开文件描述符的文件大小和修改时间信息。
- 存在的目录信息。
- 搜索文件的错误信息：文件不存在，无权限读取等错误。
- max——指定缓存的最大数量，当缓存数量不够用时，使用 LRU 规则清除最早写入、最近没有被访问过的缓存。
- inactive——指定缓存的过期时间，默认值为 60 秒。
- off——关闭缓存。

示例代码如下：

```
open_file_cache max = 1000 inactive = 20s; open_file_cache_valid 30s;
open_file_cache_min_uses 2; open_file_cache_errors on;
```

13.1.29 open_file_cache_errors 指令

语法：open_file_cache_errors on | off

默认值：open_file_cache_errors off

使用环境：http, server, location

该指令设置是否开启搜索文件的缓存错误。

13.1.30 open_file_cache_min_uses 指令

语法：open_file_cache_min_uses number

默认值：open_file_cache_min_uses 1

使用环境：http, server, location

该指令用于指定在 open_file_cache 指令设置的时间内文件的最小使用数。如果打开的文件超过该数量，则文件描述符会保持缓存中的打开状态。

13.1.31 open_file_cache_valid 指令

语法：open_file_cache_valid time

默认值：open_file_cache_valid 60

使用环境：http, server, location

该指令用于指定检查 open_file_cache 指令中条款有效性的时间，单位为秒。

13.1.32 optimize_server_names 指令

语法：optimize_server_names [on|off]

默认值：optimize_server_names on

使用环境：http, server

该指令激活或停用基于域名的虚拟主机的优化。特别地，影响了检查中使用的主机名重定向。如果打开优化选项，以及侦听所有基于域名的虚拟主机，监听一个 IP 地址和所有基于域名的服务器地址：端口对有相同的配置，然后名称不检查，并在执行请求重定向使用的第一台服务器的名称。

www.TopSage.com

如果重定向必须使用主机名并且在客户端检查通过，那么这个参数必须设置为 off。

注意：这个参数不建议在 nginx 0.7.x 版本中使用，请使用 server_name_in_redirect。

13.1.33 port_in_redirect 指令

语法：port_in_redirect [on|off]

默认值：port_in_redirect on

使用环境：http, server, location

该指令允许或阻止 Nginx 重定向过程中的端口操作。如果 port_in_redirect 为 on，Nginx 在请求重定向时则不会加上端口。

13.1.34 recursive_error_pages 指令

语法：recursive_error_pages [on|off]

默认值：recursive_error_pages off

使用环境：http, server, location

该指令允许或禁止除了第一条 error_page 指令之外的 error_page 指令。

13.1.35 resolver 指令

语法：resolver address

默认值：no

使用环境：http, server, location

13.1.36 resolver_timeout 指令

语法：resolver_timeout time

默认值：30

使用环境：http, server, location

该指令用于解析超时时间。

www.TopSage.com

大家网 TopSage.com

13.1.37 root 指令

语法：root path

默认值：root html

使用环境：http, server, location, if in location

该指令主要用来指定请求的文档根目录。例如，配置内容为 location /i/ { root /spool/w3; } 时，请求 URI 地址“/i/top.gif”将返回文件“/spool/w3/i/top.gif”的内容给客户端。

13.1.38 satisfy_any 指令

语法：satisfy_any [on|off]

默认值：satisfy_any off

使用环境：location

该指令用于检查至少一个成功的访问权限认证，它可以在 NginxHttpAccess 模块或 NginxHttpAuthBasicModule 中使用。

```
location / {
  satisfy_any  on;
  allow  192.168.1.0/32;
  deny   all;
  auth_basic            "closed site";
  auth_basic_user_file  conf/htpasswd;
}
```

13.1.39 send_timeout 指令

语法：send_timeout the time

默认值：send_timeout 60

使用环境：http, server, location

该指令用于设置发送给客户端的应答超时时间。超时时间是指进行了两次 TCP 握手，还没有转为 established 状态的时间。如果超过这个时间，客户端没有响应，Nginx 则关闭连接。

13.1.40 sendfile 指令

语法：sendfile [on|off]

www.TopSage.com

默认值：sendfile off

使用环境：http, server, location

该指令启用或禁用 sendfile()函数。sendfile()是作用于数据拷贝在两个文件描述符之间的操作函数，这个拷贝操作是在内核中操作的，所以称为“零拷贝”。sendfile 函数比 read 和 write 函数要高效得多，因为 read 和 write 要把数据拷贝到用户应用层进行操作。但是，在 Nginx 的生产环境应用中，出现过启用 sendfile 比禁用 sendfile 效率更低的情况，所以，是否启用该函数须谨慎选择。

13.1.41 server 指令

语法：server {...}

默认值：no

使用环境：http

该指令用于配置虚拟主机。在 server {......}中，使用 listen 指令来为一个虚拟主机设置监听的 IP 和端口，使用 server_name 指令来设置不同的虚拟主机名称。

13.1.42 server_name 指令

语法：server_name name [...]

默认值：server_name hostname

使用环境：server

该指令主要完成以下两项操作：

（1）根据客户端请求 Header 头信息中的 Host 域名，来匹配该请求应该由哪个虚拟主机配置 server{...}来处理。

域名处理的优先级按照以下顺序执行：

A. 全名。

B. 以通配符开头的域名，例如“*.example.com”。

C. 以通配符结尾的域名，例如“www.example.*”。

D. 使用正则表达式的域名。

如果 Header 头信息中的 Host 域名在各 server{...}中没有匹配项，该请求将按照以下优先级顺

www.TopSage.com

序由一个 server{...}来处理：

A．listen 指令被标记为 default 的 server{...}。

B．第一个出现 listen 指令（或者默认为 80 端口）的 server{...}。

（2）如果 server_name_in_redirect 设置为 on，设置的主机名将被用于 HTTP 重定向。

示例代码如下：

```
server {
  server_name   example.com  www.example.com;
}
```

第一个名称为服务器的基础名称，该虚拟主机的主机名（hostname）即为该名称。

可以使用通配符“*”来代替域名的前部分或后部分，例如：

```
server {
  server_name   example.com  *.example.com  www.example.*;
}
```

以上两个服务器名称（example.com 和*.example.com）可以写成一句：

```
server {
  server_name  .example.com;
}
```

服务器名称也可为使用正则表达式，在正则表达式加上“~”，例如：

```
server {
  server_name   www.example.com   ~^www\d+\.example\.com$;
}
```

服务器的基础名称将被用于 HTTP 重定向，如果客户端请求中没有 Host 头信息，或者 Host 头信息没有匹配 server_name，你可以使用“*”强制 Nginx 在 HTTP 重定向中使用 Host 头(注意：“*”不能作为虚拟主机的第一个名称，但是，你可以使用一个伪名称“_”来代替第一个名称)：

```
server {
  server_name example.com *;
}
server {
  server_name _ *;
}
```

注意：在 Nginx 0.6 版本中该处配置稍有不同：

```
server {
  server_name _;
}
```

从 Nginx 0.7.12 版本开始，支持空的服务器名称，用来匹配那些没有指定 Host 头信息的客户端请求：

www.TopSage.com

大家网
TopSage.com

```
server {
  server_name "";
}
```

从 Nginx 0.8.25 版本开始，可以在 server_name 正则表达式中使用命名的捕获变量（named captures），而不是数据号来获知匹配的捕获变量。

```
server {
  server_name   ~^(www\.)?(?<domain>.+)$;
  location / {
    root  /sites/$domain;
  }
}
```

一些老的 PCRE 正则表达式库提供了类似以下的语法，如果你在使用命名的捕获变量中遇到问题，可以尝试使用以下配置：

```
server {
  server_name   ~^(www\.)?(?P<domain>.+)$;
  location / {
    root  /sites/$domain;
  }
}
```

13.1.43 server_name_in_redirect 指令

语法：server_name_in_redirect on|off

默认值：server_name_in_redirect on

使用环境：http, server, location

如果 server_name_in_redirect 指令设置为 on，Nginx 将使用 server_name 指令设置的第一个名称来作重定向。如果 server_name_in_redirect 指令设置为 on，Nginx 将使用客户端请求中的 Host 头信息来作重定向。

13.1.44 server_names_hash_max_size 指令

语法：server_names_hash_max_size number

默认值：server_names_hash_max_size 512

使用环境：http

该指令用于指定服务器名称哈希表的最大值。

www.TopSage.com

13.1.45 server_names_hash_bucket_size 指令

语法：server_names_hash_bucket_size number

默认值：server_names_hash_bucket_size 32/64/128

使用环境：http

该指令用于指定服务器名称哈希表的框大小。该默认值取决于 CPU 缓存。

13.1.46 server_tokens 指令

语法：server_tokens on|off

默认值：server_tokens on

使用环境：http, server, location

是否在错误页面或服务器 Header 头中输出 Nginx 版本号给客户端浏览器。

13.1.47 tcp_nodelay 指令

语法：tcp_nodelay [on|off]

默认值：tcp_nodelay on

使用环境：http, server, location

该指令允许或禁止使用套接字选项 TCP_NODELAY，仅适用于 keep-alive 连接。

默认情况下数据发送时，内核并不会马上发送，它可能等待更多的字节组成一个包，这样可以提高 IO 发送的效率。但在每次只发送很少字节的程序中，使用 TCP_NODELAY 时等待时间就会比较长。请根据实际情况选择是否开启。

13.1.48 tcp_nopush 指令

语法：tcp_nopush [on|off]

默认值：tcp_nopush off

使用环境：http, server, location

该指令允许或禁止使用 FreeBSD 上的 TCP_NOPUSH，或者 Linux 上的 TCP_CORK 套接字

www.TopSage.com

选项。该选择仅在 sendfile 开启的时候才起作用。设置该选择的原因是 Nginx 在 Linux 和 FreeBSD 4.x 上，试图在一个包中发送它的 HTTP 应答头。

13.1.49 try_files 指令

语法：try_files param1 [param2 ... paramN] fallback

默认值：none

使用环境：location

该指令用于告诉 Nginx 测试每个文件是否存在，并且使用首先找到的文件作为 URI。如果没有找到文件，则调用 location fallback（“fallback”可以为任何名称）。fallback 是一个请求参数，它可以是一个命名的 location，也可以是任何可能的 URI。示例：

```
location / {
  try_files index.html index.htm @fallback;
}

location @fallback {
  root /var/www/error;
  index index.html;
}
```

13.1.50 types 指令

语法：types {...}

使用环境：http, server, location

该指令用于应答的 MIME-types 对应的扩展名，一个 MIME-types 可以对应多个扩展名。默认情况下，使用这些扩展名：

```
types {
  text/html   html;
  image/gif   gif;
  image/jpeg  jpg;
}
```

完整的 MIME-types 与扩展名的映射表在被包含的 conf/mime.types 文件中。

如果你想让某些 location 使用 MIME 类型：application/octet-stream，则可以使用以下配置：

```
location /download/ {
  types        { }
  default_type application/octet-stream;
}
```


www.TopSage.com

大家网 TopSage.com

13.1.51 HTTP 核心模块中可以使用的变量

Nginx HTTP 核心模块支持一些与 Apache 变量名称相同的内置变量，例如：$http_user_agent、$http_cookie，此外，还支持一些 Nginx 特有的其他变量：

$arg_PARAMETER

该变量包含了当查询字符串时，GET 请求可变参数的值。

$args

这个变量等于请求行中的参数。

$binary_remote_addr

二进制格式的客户端地址。

$content_length

这个变量等于客户端请求头中的 content-length 值。

$content_type

这个变量等于客户端请求头中的 content-type 值。

$cookie_COOKIE

客户端请求 Header 头中的 cookie 变量。前缀“$cookie_”加上 cookie 名称的变量，该变量的值即为 cookie 名称的值。例如客户端请求头中某一行的内容为“Cookie:PHPSESSID=6fd171f5c1b4e42783f2b6b9d7124065; userid=2”，那么 Nginx 变量$cookie_userid 的值则为 2。

$document_root

这个变量等于当前请求所属的 root 指令设置的文档根目录路径。

$document_uri

这个变量与$uri 类似。

$host

这个变量等于客户端请求头中的 Host 值，如果客户端请求头中没有 Host 值，那么这个变量等于为当前请求提供服务的服务器名称。

$http_HEADER

客户端请求 header 头中的变量。将客户端请求头每行名称中的横线“-”换成下划线“_”，

www.TopSage.com

大写字母转为小写字母，前缀加上“$http_”，即为该名称对应的变量。例如客户端请求头中 cookie 行的内容为“Accept-Encoding: gzip, deflate”，那么在 Nginx 中则有一个变量 $http_accept-encoding，该变量的值为“gzip, deflate”。类似的变量有：$http_user_agent、$http_referer...

$is_args

如果$args 已经设置，则该变量的值为"?"，否则为""。

$limit_rate

这个变量允许限制连接速率。

$query_string

这个与$args 类似。

$remote_addr

客户端的 IP 地址。

$remote_port

客户端的端口。

$remote_user

这个变量等于用户名，Auth Basic 模块中会使用到该变量。

$request_filename

这个变量等于当前请求的文件路径，由 root、alias 指令及 URI 请求生成。

$request_body

这个变量（0.7.58 以上版本支持）包含了请求的 body 主体内容。在使用 proxy_pass 或 fastcgi_pass 指令的 location 中，这个变量比较有意义。

$request_body_file

客户端请求主体的临时文件名。

$request_method

这个变量等于 HTTP 请求的动作，常使用的动作为 GET 或 POST。

www.TopSage.com

在 0.8.20 及之前的版本中，这个变量总是等于主请求的动作名，如果当前请求是一个子请求，该变量的值并不是当前请求的动作名。

$request_uri

这个变量等于带有参数的完整 URI。

$scheme

HTTP 方法（例如 IE 浏览器支持的 http 和 https）。按需使用，示例如下：

```
rewrite ^(.+)$ $scheme://example.com$1 redirect;
$server_addr
```

这个变量等于服务器的地址。通常，在一次系统调用后，该变量的值可以确定。如果为了避开系统调用，则要在 listen 指令中指出地址，并且使用参数 bind。

$server_name

服务器的主机名。

$server_port

请求到达的服务器端口。

$server_protocol

这个变量等于请求采用的协议，通常该变量的值为 HTTP/1.0 或 HTTP/1.1。

$uri

这个变量等于当前请求的 URI（不带参数），它可以不同于最初的值，例如可以被内部重定向所改变。

13.2 HTTP Upstream 模块

HTTP Upstream 模块是与反向代理、负载均衡相关的模块，包含 ip_hash、server、upstream 等指令。这一模块已在前面第 6 章中做了详细介绍。

13.3 HTTP Access 模块

HTTP Access 模块提供了一个简单的基于 host 名称的访问控制。通过该模块，可以允许或禁止指定的 IP 地址或 IP 段访问某些虚拟主机或目录。

www.TopSage.com

大家网 TopSage.com

示例如下：

```
location / {
  deny    192.168.1.1;
  allow   192.168.1.0/24;
  allow   10.1.1.0/16;
  deny    all;
}
```

13.3.1 allow 指令

语法：allow [address | CIDR | all]

默认值：none

使用环境：http, server, location, limit_except

允许指定的 IP 地址或 IP 段访问某些虚拟主机或目录。

13.3.2 deny 指令

语法：deny [address | CIDR | all]

默认值：none

使用环境：http, server, location, limit_except

禁止指定的 IP 地址或 IP 段访问某些虚拟主机或目录。

13.4 HTTP Auth Basic 模块

该模块采用基于HTTP基本身份验证的用户名和密码登录方式，来保护你的虚拟主机或目录。示例如下：

```
location / {
  auth_basic            "Restricted";
  auth_basic_user_file  htpasswd;
}
```

13.4.1 auth_basic 指令

语法：auth_basic [text|off]

默认值：auth_basic off

www.TopSage.com

使用环境：http, server, location, limit_except

该指令用于指定弹出的用户名和密码登录框中提示的名称，例如设置 auth_basic 名称为“Kingsoft”，效果如图 13-1 所示：

```
location /admin/ {
   auth_basic           "Kingsoft";
   auth_basic_user_file  htpasswd;
}
```

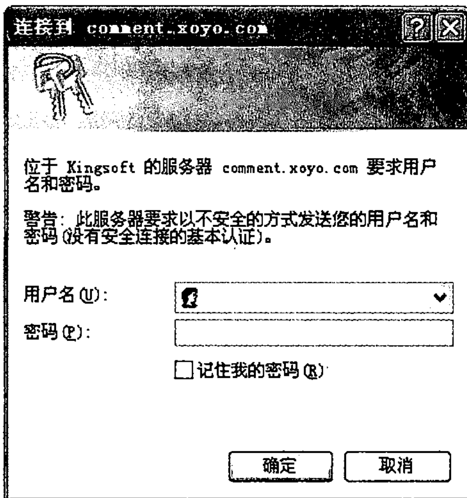


图 13-1 auth_basic 指令弹出的用户名和密码登录框

如果设置 auth_basic off，则关闭 HTTP 基本身份验证。

13.4.2 auth_basic_user_file 指令

语法：auth_basic_user_file the_file

默认值：no

使用环境：http, server, location, limit_except

该指令用于设置 htpasswd 密码文件的名称。在 Nginx 0.6.7 版本之前，htpasswd 密码文件相对路径为 Nginx 安装目录，在 Nginx 0.6.7 版本之后，htpasswd 密码文件的相对路径为 nginx.conf 文件所在的路径。如果您的 Nginx 安装在/usr/local/nginx/路径下，并设置了“auth_basic_user_file htpasswd;”，那么 Nginx 将读取/usr/local/nginx/conf/htpasswd 文件。

htpasswd 密码文件的格式如下：

www.TopSage.com

```
用户名:密码
用户名2:密码2:注释
用户名3:密码3[1]
```

你也可以使用 perl 创建密码文件，pw.pl 的内容如下：

```
#!/usr/bin/perl
use strict;

my $pw=$ARGV[0] ;
print crypt($pw,$pw)."\n";
```

然后执行：

```
chmod +x pw.pl
./pw.pl password
papAq5PwY/QQM
```

papAq5PwY/QQM 就是 password 的 crypt()密码。

13.5 HTTP Autoindex 模块

该模块用于提供自动目录列表。该模块只有在找不到默认的 index 文件时才启用。

13.5.1 autoindex 指令

语法：autoindex [on|off]

默认值：autoindex off

使用环境：http, server, location

该指令允许和禁止自动目录列表。

13.5.2 autoindex_exact_size 指令

语法：autoindex_exact_size [on|off]

默认值：autoindex_exact_size on

使用环境：http, server, location

该指令用于设置按照什么单位显示目录列表中文件的大小、默认字节，或者 KB、MB、GB。

[1] 密码必须使用函数 crypt（3）加密，你可以通过 Apache 的 htpasswd 程序生成密码。

www.TopSage.com



13.5.3 autoindex_localtime 指令

语法：autoindex_localtime [on|off]

默认值：autoindex_localtime off

使用环境：http, server, location

该指令的参数为 on 时，采用本地时间显示文件修改时间，为 off 时，采用 GMT 格林尼治时间显示文件修改时间。

13.6 HTTP Browser 模块

该模块可以根据客户端“User-agent”请求头中的信息创建变量。

$modern_browser——如果浏览器被识别为新浏览器，则等于 modern_browser_value 指令设置的值。

$ancient_browser——如果浏览器被识别为旧浏览器，则等于 ancient_browser_value 指令设置的值。

$msie——如果浏览器被鉴定为 MSIE 浏览器，该变量的值则为 1。

配置示例如下：

（1）根据浏览器类型选择不同的首页文件，代码如下：

```
modern_browser_value "modern.";
modern_browser msie 5.5;
modern_browser gecko 1.0.0;
modern_browser opera 9.0;
modern_browser safari 413;
modern_browser konqueror 3.0;
index index.${modern_browser}html index.html;
```

（2）重定向旧的浏览器，代码如下：

```
modern_browser msie 5.0;
modern_browser gecko 0.9.1;
modern_browser opera 8.0;
modern_browser safari 413;
modern_browser konqueror 3.0;
modern_browser unlisted;
ancient_browser Links Lynx Netscape4;
```


www.TopSage.com

```
if ($ancient_browser){
  rewrite ^ /ancient.html;
}
```

13.6.1 ancient_browser 指令

语法：ancient_browser line [line...]

默认值：no

使用环境：http, server, location

该指令用于设置“User-agent”头信息中包含的旧的浏览器名称，多个浏览器名称可以用空格分隔。特殊的行可以采用正则表达式，例如“^Mozilla/[1-4]”。

13.6.2 ancient_browser_value 指令

语法：ancient_browser_value line

默认值：ancient_browser_value 1

使用环境：http, server, location

该指令为$ancient_browser 变量指定 Value 值。

13.6.3 modern_browser 指令

语法：modern_browser browser version|unlisted

默认值：no

使用环境：http, server, location

该指令用于设置新浏览器类型，可选的浏览器参数只能是 msie、gecko （Mozilla-based 浏览器）opera、safari、konqueror 几种。

版本可以设置为 X、X.X、X.X.X 或 X.X.X.X，它们各自的最大值分别为 4000、4000.99、4000.99.99 和 4000.99.99.99。

特殊的值“unlisted”表示即不属于新浏览器，也不属于旧浏览器的其他浏览器类型。

如果“User-Agent”头信息为空值，将会被归为新浏览器。

代码 13-3 的配置可以判断客户端浏览器是否为手机浏览器：

www.TopSage.com

代码 13-3

```
modern_browser unlisted;

ancient_browser "GoBrowser";
ancient_browser "MIDP";
ancient_browser "WAP";
ancient_browser "UP.Browser";
ancient_browser "Smartphone";
ancient_browser "Obigo";
ancient_browser "Mobile";
ancient_browser "AU.Browser";
ancient_browser "wxd.Mms";
ancient_browser "WxdB.Browser";
ancient_browser "CLDC";
ancient_browser "UP.Link";
ancient_browser "KM.Browser";
ancient_browser "UCWEB";
ancient_browser "SEMC-Browser";
ancient_browser "Mini";
ancient_browser "Symbian";
ancient_browser "Palm";
ancient_browser "Nokia";
ancient_browser "Panasonic";
ancient_browser "MOT-";
ancient_browser "SonyEricsson";
ancient_browser "NEC-";
ancient_browser "Alcatel";
ancient_browser "Ericsson";
ancient_browser "BENQ";
ancient_browser "BenQ";
ancient_browser "Amoisonic";
ancient_browser "Amoi";
ancient_browser "Capitel";
ancient_browser "PHILIPS";
ancient_browser "SAMSUNG";
ancient_browser "Lenovo";
ancient_browser "Mitsu";
ancient_browser "Motorola";
ancient_browser "SHARP";
ancient_browser "WAPPER";
ancient_browser "LG-";
ancient_browser "LG/";
ancient_browser "EG900";
ancient_browser "CECT";
ancient_browser "Compal";
ancient_browser "kejian";
ancient_browser "Bird";
ancient_browser "BIRD";
ancient_browser "G900/V1.0";
ancient_browser "Arima";
ancient_browser "CTL";
```


www.TopSage.com

大家网 TopSage.com

```
ancient_browser "TDG";
ancient_browser "Daxian";
ancient_browser "DBTEL";
ancient_browser "Eastcom";
ancient_browser "EASTCOM";
ancient_browser "PANTECH";
ancient_browser "Dopod";
ancient_browser "Haier";
ancient_browser "HAIER";
ancient_browser "KONKA";
ancient_browser "KEJIAN";
ancient_browser "LENOVO";
ancient_browser "Soutec";
ancient_browser "SOUTEC";
ancient_browser "SAGEM";
ancient_browser "SEC";
ancient_browser "SED-";
ancient_browser "EMOL";
ancient_browser "INNO55";
ancient_browser "ZTE";
ancient_browser "iPhone";
ancient_browser "Android";
ancient_browser "Windows CE";
ancient_browser "DX";
ancient_browser "TELSON";
ancient_browser "TCL";
ancient_browser "oppo";
ancient_browser "ChangHong";
ancient_browser "MALATA";
ancient_browser "KTOUCH";
ancient_browser "TIANYU";
ancient_browser "TOUCH";
ancient_browser "MAUI";
ancient_browser "J2ME";
ancient_browser "BlackBerry";
ancient_browser "yulong";
ancient_browser "coolpad";

if ( $ancient_browser )
{
   proxy_pass http://mobile.domain.com;
}
```

13.6.4 modern_browser_value 指令

语法：modern_browser_value line

默认值：modern_browser_value 1

使用环境：http, server, location

www.TopSage.com

该变量为$modern_browser 变量指定 Value 值。

13.7 HTTP Charset 模块

该模块用来添加文本编码类型到 HTTP 应答头“Content-Type indicated”。此外，该模块还能够将服务器端网页原来的文本编码转换成另一种文本编码，输出给客户端。例如：

```
charset        windows-1251;
source_charset  koi8-r;
```

13.7.1 charset 指令

语法：charset encoding|off

默认值：charset off

使用环境：http, server, location, if in location

该指令用来添加文本编码类型到 HTTP 应答头“Content-Type indicated”。如果编码与 source_charset 指令设置的编码不一样，将进行重编码。参数为“off”。

13.7.2 charset_map 指令

语法：charset_map encoding1 encoding2 {...}

默认值：no

使用环境：http, server, location

该指令用来添加字符集映射，可以将一种编码映射成另一种编码。代码符号以十六进制方式表示。该指令主要是针对俄文的，示例如下：

```
charset_map  koi8-r  windows-1251 {
  C0  FE  ; # small yu
  C1  E0  ; # small a
  C2  E1  ; # small b
  C3  F6  ; # small ts
  # ...
}
```

以上示例是为了将 KOI8-R 编码转换成 Windows-1251 编码。

美国英语使用的编码是 ASCII （American National Standard Code）编码，它是一种 7 位（128 个字符）编码，只包含拉丁字母和一些符号，因此西里尔字符编码采用 7 位是不够的。KOI8-R

www.TopSage.com

（Kod Obmena Informatsii，俄语 Код Обмена Инфармации 的拉丁写法，缩写为 КОИ8）是一种官方的 Internet 标准编码，它是一种包含西里尔字母的 8 位编码，它是由 KOI8 派生的，KOI8-R 中的 8 就是指 8 位的意思，R 指俄语（KOI8-U 使用于乌克兰语），这在 Cyrillic Character Set （RFC, Request For Comments, 请求注解，Internet 标准草案，1489）的注册纪录中有详细描述。它同时也是事实上的 e-mail 和 NNTP （Network News Transfer Protocol, 网络新闻传输协议）的西里尔字符标准。另外，它还是 Unix 系统的标准编码，如图 13-2 所示。

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
─	│	┌	┐	└	┘	├	┤	┬	┴	┼	▀	▄	█	▌	▐
█	█	█		·	•	√	≈	≤	≥			°	²	·	÷
═	║	╒	ё	╓	╔	╕	╖	╗	╘	╙	╚	╛	╜	╝	╞
╟	╠	╡	Ё	╢	╣	╤	╥	╦	╧	╨	╩	╪	╫	╬	©
ю	а	б	ц	д	е	ф	г	х	и	SPC	к	л	м	н	о
п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

图 13-2 KOI8-R 编码

CP1251 是 MS Widows 编码页 1251，它是 Microsoft 西里尔字符的标准编码，它也恰恰是 MS Windows 平台实事上的西里尔字符标准，如图 13-3 所示。

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
Ђ	Ѓ	‚	ѓ	„	…	†	‡		‰	Љ	‹	Њ	Ќ	Ћ	Џ
ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
	Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
А	Б	В	Г	Д	Е	Ж	З	И	Й	SPC	Л	М	Н	О	П
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

图 13-3 Widows-CP1251 编码

www.TopSage.com

大家网
TopSage.com

13.7.3 override_charset 指令

语法：override_charset on|off

默认值：override_charset off

使用环境：http, server, location, if in location

当该指令开启时，如果后端的 FASTCGI 服务器响应头带有“Content-Type”头信息，将开启编码转换。

13.7.4 source_charset 指令

语法：source_charset encoding

默认值：no

使用环境：http, server, location, if in location

该指令用于设定原始编码，如果编码与 charset 指令设置的编码不一样，将进行重编码。

13.8 HTTP Empty Gif 模块

该指令可以保持一个 1×1 像素的透明 GIF 图片在内存中，当请求该图片时，能够得到非常快的响应速度。示例如下：

```
location = /_.gif {
  empty_gif;
}
```

13.8.1 empty_gif 指令

语法：empty_gif

默认值：n/a

使用环境：location

13.9 HTTP Fcgi 模块

该模块用于设置 Nginx 与 FastCGI 进程交互，并通过传递参数来控制 FastCGI 进程工作。

www.TopSage.com

示例如代码 13-4 所示：

代码 13-4

```
location / {
  fastcgi_pass   localhost:9000;
  fastcgi_index  index.php;

  fastcgi_param  SCRIPT_FILENAME  /home/www/scripts/php$fastcgi_script_name;
  fastcgi_param  QUERY_STRING    $query_string;
  fastcgi_param  REQUEST_METHOD   $request_method;
  fastcgi_param  CONTENT_TYPE    $content_type;
  fastcgi_param  CONTENT_LENGTH   $content_length;
}
```

缓冲示例如代码 13-5 所示：

代码 13-5

```
http {
  fastcgi_cache_path   /path/to/cache  levels=1:2
                       keys_zone=NAME:10m
                       inactive=5m    clean_time=2h00m;

  server {
    location / {
      fastcgi_pass    http://127.0.0.1;
      fastcgi_cache   NAME;
      fastcgi_cache_valid   200 302  1h;
      fastcgi_cache_valid   301      1d;
      fastcgi_cache_valid   any      1m;
      fastcgi_cache_min_uses  1;
      fastcgi_cache_use_stale error  timeout invalid_header http_500;
    }
  }
}
```

13.9.1 fastcgi_buffers 指令

语法：fastcgi_buffers the_number is_size;

默认值：fastcgi_buffers 8 4k/8k;

使用环境：http, server, location

该指令设置了读取 FastCGI 进程返回信息的缓冲区数量和大小。默认情况下，一个缓冲区的大小应该跟操作系统的页大小一样，默认为 4k/8k 自动选择。

你可以通过以下命令查看系统页大小：

www.TopSage.com

```
getconf PAGESIZE
```

或

```
getconf PAGE_SIZE
```

或者编译以下 C 代码来查看：

```
#include <unistd.h>
#include <stdio.h>
int main()
{
 printf("Page size on your system = %i bytes\n", getpagesize());
 return 0;
}
```

13.9.2 fastcgi_buffer_size 指令

语法：fastcgi_buffer_size the_size

默认值：fastcgi_buffer_size 4k/8k

使用环境：http, server, location

该指令设置 FastCGI 服务器相应头部的缓冲区大小。通常情况下，该缓冲区的大小设置等于 fastcgi_buffers 指令设置的一个缓冲区的大小。默认为 4k/8k 自动选择。

13.9.3 fastcgi_cache 指令

语法：fastcgi_cache zone;

默认值：off

使用环境：http, server, location

该指令用来设置缓存在共享内存中的名称。一款区域可能被几个地方引用。

13.9.4 fastcgi_cache_key 指令

语法：fastcgi_cache_key line ;

默认值：none

使用环境：http, server, location

该指令用来设置被缓存的 key，示例如下：

```
fastcgi_cache_key localhost: 9000 $ request_uri;
```


www.TopSage.com

大学网 TopSage.com

13.9.5 fastcgi_cache_methods 指令

语法：fastcgi_cache_methods [GET HEAD POST];

默认值：fastcgi_cache_methods GET HEAD;

使用环境：main, http, location

该指令用于设置哪些 HTTP 请求可以被缓存。参数可选值为 GET、HEAD、POST，默认值为 GET、HEAD。另外，如果你只设置“fastcgi_cache_methods POST;”，GET/HEAD 的缓存不会被禁止。

13.9.6 fastcgi_index 指令

语法：fastcgi_index file

默认值：none

使用环境：http, server, location

如果请求的 FastCGI URI 以“/”斜线结束，该指令设置的文件会被附加到 URI 的后面并保存在变量$fastcgi_script_name 中。

13.9.7 fastcgi_hide_header 指令

语法：fastcgi_hide_header name

使用环境：http, server, location

默认情况下，Nginx 不会将 FastCGI 进程返回的“Status”、“X-Accel-...”Header 头信息返回给客户端。该指令可以被用来隐藏其他的 Header 头信息。

如果需要“Status”、“X-Accel-...”消息头，就须要使用 fastcgi_pass_header 指令让 FastCGI 强制发送消息头给客户端。

13.9.8 fastcgi_ignore_client_abort 指令

语法：fastcgi_ignore_client_abort on|off

默认值：fastcgi_ignore_client_abort off

www.TopSage.com

使用环境：http, server, location

如果客户端中断对服务器的请求，则该指令可决定当前对 FastCGI 的请求是否该中断。

13.9.9 fastcgi_intercept_errors 指令

语法：fastcgi_intercept_errors on|off

默认值：fastcgi_intercept_errors off

使用环境：http, server, location

该指令用来决定是否要把客户端转向 4xx 和 5xx 错误页，或者允许 Nginx 自动指定错误页。

注意：你需要在此明确错误页，它才是有用的。Igor 曾说："如果没有定制的处理机制，Nginx 不会拦截一个没有默认页的错误。Nginx 只会拦截一些小的错误，放过一些其他错误。

13.9.10 fastcgi_max_temp_file_size 指令

语法：fastcgi_max_temp_file_size 0

该指令用于关闭磁盘缓冲。

13.9.11 fastcgi_param 指令

语法：fastcgi_param parameter value

默认值：none

使用环境：http, server, location

该指令指定的参数，将被传递给 FastCGI-server。

它可能使用字符串、变量及其它们的组合来作为参数值。如果不在此制定参数，它就会继承外层设置；如果在此设置了参数，将清除外层相关设置，仅启用本层设置。

下面是一个例子，对 PHP 来说的最精简的必要参数：

```
fastcgi_param  SCRIPT_FILENAME  /home/www/scripts/php$fastcgi_script_name;
fastcgi_param  QUERY_STRING     $query_string;
```

参数 SCRIPT_FILENAME 是 PHP 用来确定执行脚本的名字，而参数 QUERY_STRING 是它的一个子参数。

www.TopSage.com

如果要处理 POST，那么这三个附加参数是必要的：

```
fastcgi_param  REQUEST_METHOD   $request_method;
fastcgi_param  CONTENT_TYPE     $content_type;
fastcgi_param  CONTENT_LENGTH   $content_length;
```

如果 PHP 在编译时使用了——enable-force-cgi-redirect 选项，设置参数 REDIRECT_STATUS 的值为 200 就是必须的了。

```
fastcgi_param  REDIRECT_STATUS  200;
```

13.9.12 fastcgi_pass 指令

语法：fastcgi_pass fastcgi-server

默认值：none

使用环境：location, if in location

该指令用于指定 FastCGI 服务器监听的端口或 Unix 套接字。

示例如下：

```
fastcgi_pass   localhost:9000;
```

使用 Unix 套接字的示例如下：

```
fastcgi_pass   unix:/tmp/fastcgi.socket;
```

如果有多台 FastCGI 服务器，你也可以使用 upstream 指令指定 FastCGI 服务器，示例如下：

```
upstream backend {
  server   localhost:1234;
}
fastcgi_pass   backend;
```

13.9.13 fastcgi_pass_header 指令

语法：fastcgi_pass_header name

默认值：http, server, location

13.9.14 fastcgi_read_timeout 指令

语法：fastcgi_read_timeout time

www.TopSage.com



默认值：60

使用环境：http, server, location

该指令用于设置 upstream 模块等待 FastCGI 进程发送数据的超时时间，默认值为 60 秒，如果你有执行时间较长的 FastCGI 程序，则须更改此值。

13.9.15 fastcgi_redirect_errors 指令

语法：fastcgi_redirect_errors on|off

该指令用于开启或关闭 FastCGI 错误重定向。

13.9.16 fastcgi_split_path_info 指令

语法：fastcgi_split_path_info regex

使用环境：location

Nginx 版本要求：>= 0.7.31

```
location ~ ^(.+\.php)(.*)$ {
...
fastcgi_split_path_info ^(.+\.php)(.*)$;
fastcgi_param SCRIPT_FILENAME /path/to/php$fastcgi_script_name;
fastcgi_param PATH_INFO $fastcgi_path_info;
fastcgi_param PATH_TRANSLATED $document_root$fastcgi_path_info;
...
}
```

13.10 geo 模块

geo 模块主要用于做全局负载均衡，可以根据不同的客户端 IP 访问到不同的服务器。一些针对不同地区的客户，使用不同的服务器去处理的需求，可以使用 geo 模块。

geo 简单的配置示例如下：

```
geo  $geo  {
 default         0;
 127.0.0.1/32     2;
 192.168.1.0/24   1;
 10.1.0.0/16      1;
}
```

下面，我们来看 geo 全局负载均衡与 upstream 反向代理的配合使用，示例如代码 13-6 所示：

www.TopSage.com

代码 13-6

```
user  www www;

worker_processes 10;

error_log  /data1/logs/nginx_error.log  crit;

pid        /tmp/nginx.pid;

worker_rlimit_nofile 51200;

events
{
    use epoll;

    worker_connections 51200;
}

http
{
    include       conf/mime.types;
    default_type  application/octet-stream;

    geo $geo {
        default             default;
        218.30.115.0/24       china_telecom;
        202.106.182.0/24      china_unicom;
        202.205.3.0/24        cernet;
    }

    upstream   default.server {
        server 192.168.0.2;
    }

    upstream   china_telecom.server {
        server 192.168.0.3;
    }

    upstream   china_unicom.server {
        server 192.168.0.4;
    }

    upstream   cernet.server {
        server 192.168.0.5;
    }

    server {
        listen      80;
        server_name  www.yourdomain.com;
        index index.html index.htm index.php;
        location / {
```


www.TopSage.com

```
            proxy_redirect off;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://$geo.server$request_uri;
        }
    }
}
```

13.10.1 geo 指令

语法：geo [$ip_variable] $variable { ... }

默认值：none

使用环境：http

geo 指令默认使用客户端的 IP 地址（变量$remote_addr 的值）来进行全局负载均衡，从 Nginx 0.7.27 版本开始，可以支持使用指定的变量来做全局负载均衡，例如：

```
geo  $arg_remote_addr $geo {
  ...;
}
```

geo 指令支持以下描述：

default：任何 IP 地址，相当于 0.0.0.0/0。

include：可以引用一个文本文件，里面包含 geo 的配置内容。当 geo 配置的 IP 段较多时，可以将 IP 段写在另一个配置文件中，并引用进来。

ranges：从 Nginx 0.7.23 版本开始，支持使用区间形式来指定 IP 段。该指令必须写在 geo 配置环境的第一行。

Default、include 描述示例代码如 13-7 所示：

代码 13-7

```
geo  $country  {
  default          no;
  include          conf/geo.conf;
  127.0.0.0/24     us;
  127.0.0.1/32     ru;
  10.1.0.0/16      ru;
  192.168.1.0/24   uk;
}
```

引用的配置文件 conf/geo.conf 的内容如下：

www.TopSage.com

大家网 TopSage.com

```
10.2.0.0/16      ru;
192.168.2.0/24   ru;
```

当指定的 IP 段区间存在包含关系时，取最精确的配值。例如在以下示例中，IP 地址 127.0.0.1 取得的 value 值为 ru，而不是 us。

Ranges 区间形式指定 IP 段示例如代码 13-8 所示：

代码 13-8

```
geo $country {
  ranges;
  default                    no;
  127.0.0.0-127.0.0.0        us;
  127.0.0.1-127.0.0.1        ru;
  127.0.0.1-127.0.0.255      us;
  10.1.0.0-10.1.255.255      ru;
  192.168.1.0-192.168.1.255  uk;
}
```

13.11 Gzip 模块

Gzip 模块主要用于对返回给客户端的网页采用 gzip 进行压缩输出。

目前，90%的浏览器都支持 gzip 和 deflate 两种压缩格式。如果浏览器支持 gzip 压缩，就会在 HTTP 请求头中发送一行"Accept-Encoding: gzip, deflate"，这时候 Nginx 服务器可以输出经过 gzip 压缩后的页面给浏览器，浏览器再解压。这种方式可以将网络线路上传输的大量数据消减 60% 以上，不仅节省了服务器带宽，同时加速了用户的下载速度和体验。

一个典型的 IE7 浏览器 HTTP GET 请求头的信息如代码 13-9 所示：

代码 13-9

```
GET / HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, application/x-ms-application,
application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xaml+xml, */*
Accept-Language: zh-cn
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0;
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) ; baiduds; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; CIBA)
Accept-Encoding: gzip, deflate
Host: www.xoyo.com
Connection: Keep-Alive
```


www.TopSage.com

13.11.1 gzip 指令

语法：gzip on|off

默认值：gzip off

使用环境：http, server, location, if （x） location

该指令用于开启或关闭 gzip 模块。

13.11.2 gzip_buffers 指令

语法：gzip_buffers number size

默认值：gzip_buffers 4 4k/8k

使用环境：http, server, location

设置系统获取几个单位的缓存用于存储 gzip 的压缩结果数据流。例如 4 4k 代表以 4k 为单位，按照原始数据大小以 4k 为单位的 4 倍申请内存。 4 8k 代表以 8k 为单位，按照原始数据大小以 8k 为单位的 4 倍申请内存。

如果没有设置，默认值是申请跟原始数据相同大小的内存空间去存储 gzip 压缩结果。

13.11.3 gzip_comp_level 指令

语法：gzip_comp_level 1..9

默认值：zip_comp_level 1

使用环境：http, server, location

gzip 压缩比，1 压缩比最小处理速度最快，9 压缩比最大但处理速度最慢（传输快但比较消耗 cpu）。

13.11.4 gzip_min_length 指令

语法：gzip_min_length length

默认值：gzip_min_length 0

使用环境：http, server, location

www.TopSage.com

设置允许压缩的页面最小字节数，页面字节数从 header 头的 Content-Length 中进行获取。默认值是 0，不管页面多大都压缩。建议设置成大于 1k 的字节数，小于 1k 可能会越压越大。即：gzip_min_length 1024。

13.11.5 gzip_http_version 指令

语法：gzip_http_version 1.0|1.1

默认值：gzip_http_version 1.1

使用环境：http, server, location

识别 http 的协议版本。由于早期的一些浏览器或 http 客户端，可能不支持 gzip 自解压，用户就会看到乱码，所以做一些判断还是有必要的[2]。

13.11.6 gzip_proxied 指令

语法：gzip_proxied [off|expired|no-cache|no-store|private|no_last_modified|no_etag|auth|any] ...

默认值：gzip_proxied off

使用环境：http, server, location

Nginx 作为反向代理的时候启用，开启或关闭后端服务器返回的结果，匹配的前提是后端服务器必须要返回包含“Via”的 header 头。

off——关闭所有的代理结果数据的压缩。

expired——启用压缩，如果 header 头中包含“Expires”头信息。

no-cache——启用压缩，如果 header 头中包含“Cache-Control:no-cache”头信息。

no-store——启用压缩，如果 header 头中包含“Cache-Control:no-store”头信息。

private——启用压缩，如果 header 头中包含“Cache-Control:private”头信息。

no_last_modified——启用压缩，如果 header 头中不包含“Last-Modified”头信息。

no_etag——启用压缩，如果 header 头中不包含“ETag”头信息。

auth——启用压缩，如果 header 头中包含“Authorization”头信息。

[2] 21 世纪都来了，现在除了类似于百度的蜘蛛之类的东西不支持自解压，99.99%的浏览器基本上都支持 gzip 解压，所以可以不用设这个值，保持系统默认即可。

www.TopSage.com

大家网 TopSage.com

any——无条件启用压缩。

13.11.7 gzip_types 指令

语法：gzip_types mime-type [mime-type ...]

默认值：gzip_types text/html

使用环境：http, server, location

匹配 mime 类型进行压缩，（无论是否指定）“text/html”类型总是会被压缩的。

注意：如果作为 http server 来使用，主配置文件中要包含文件类型配置文件。

```
http
{
  include      conf/mime.types;
  ......
}
```

如果你希望压缩常规的文件类型，可以写成代码 13-10 这样。

代码 13-10

```
http
{
 include      conf/mime.types;

 gzip on;
 gzip_min_length  1000;
 gzip_buffers     4 8k;
 gzip_http_version 1.1;
 gzip_types      text/plain application/x-javascript text/css text/html
   application/xml;

 ......
}
```

13.12 HTTP Headers 模块

这一组指令主要用来设置 Nginx 返回网页内容给用户时，附加的 Header 头信息。示例：

```
expires     24h;
expires     0;
expires     -1;
expires     epoch;
add_header   Cache-Control private;
```


www.TopSage.com

13.12.1 add_header 指令

语法：add_header name value

默认值：none

使用环境：http, server, location

当 HTTP 应答状态码为 200、204、301、302 或 304 的时候，增加指定的 Header 头，其中 Header 头的值可以使用变量。

13.12.2 expires 指令

语法：expires [time|epoch|max|off]

默认值：expires off

使用环境：http, server, location

使用本指令可以控制 HTTP 应答中的"Expires"和"Cache-Control" Header 头信息，起到控制浏览器端页面缓存的作用。

可以在 time 值中使用正数或负数。"Expires"头标的值将通过当前系统时间加上您设定的 time 值来获得。

epoch 指定"Expires"的值为 1 January, 1970, 00:00:01 GMT。

max 指定"Expires"的值为 31 December 2037 23:59:59 GMT，"Cache-Control"的值为 10 年。

-1 指定"Expires"的值为服务器当前时间-1s，即永远过期。

"Cache-Control"头标的值由您指定的时间来决定：

负数：Cache-Control: no-cache。

正数或零：Cache-Control: max-age = #, # 为您指定时间的秒数。

"off"表示不修改"Expires"和"Cache-Control"的值。

注意：expires 指令同样也只能工作于 200、204、301、302 或 304 应答状态。

13.13 HTTP Index 模块

该模块可以用于指定虚拟主机目录下的默认首页文件名称。例如：

www.TopSage.com

```
index index.html;
```

也可以指定多个默认首页文件名称，如果您指定了多个文件，那么将按照从左到右的顺序逐个查找。可以在列表末尾加上一个绝对路径名的文件。例如：

```
index index.html index.htm;
```

13.13.1 index 指令

语法：index file-path [file-path [...]];

默认值：no

使用环境：server, location

该指令用来指定做默认文档的文件名，可以在文件名处使用变量。如果您指定了多个文件，那么将按照您指定的顺序逐个查找。可以在列表末尾加上一个绝对路径名的文件。

示例如下：

```
index  index.$geo.html  index.0.html  /index.html;
```

13.14 HTTP Referer 模块

HTTP Referer 是 Header 的一部分，当浏览器向 Web 服务器发送请求的时候，一般会带上 Referer，告诉服务器我是从哪个页面链接过来的，服务器借此可以获得一些信息用于处理，例如防止未经允许的网站盗链图片、文件等。因为 HTTP Referer 头信息是可以通过程序来伪装生成的，所以，通过 Referer 信息防盗链并非 100%可靠，但是，它能够限制大部分的盗链情况。示例如代码 13-11 所示：

代码 13-11

```
location /photos/ {
  valid_referers none blocked www.mydomain.com mydomain.com;

  if ($invalid_referer) {
    return  403;
  }
}
```

13.14.1 valid_referers 指令

语法：valid_referers [none|blocked|server_names] ...

www.TopSage.com

大家网 TopSage.com

默认值：none

使用环境：server, location

该指令会根据 Referer Header 头的内容分配一个值 0 或 1 给变量$invalid_referer。如果 Referer Header 头不符合 valid_referers 指令设置的有效 Referer，变量$invalid_referer 将被设置为 1（详情请见代码 13-11 示例）。

该指令的参数可以为以下内容：

none：表示无 Referer 值的情况；

blocked：表示 Referer 值被防火墙进行伪装，例如："Referer: XXXXXXX"；

server_names：表示一个或多个主机名称。从 Nginx 0.5.33 版本开始，server_names 中可以使用通配符"*"号。

13.15 HTTP Limit Zone 模块

该模块用于针对条件，进行会话的并发连接数控制，例如限制每个 IP 的并发连接数等。配置示例如代码 13-12 所示：

代码 13-12

```
http {
 limit_zone  one $binary_remote_addr 10m;

 server {
  location /download/ {
   limit_conn  one 1;
  }
 }
}
```

13.15.1 limit_zone 指令

语法：limit_zone zone_name $variable memory_max_size

默认值：no

使用环境：http

该指令定义了一个数据区，其中记录会话状态信息。$variable 定义判断会话的变量；memory_max_size 定义内存记录区的总容量。示例如下：

www.TopSage.com

```
limit_zone one $binary_remote_addr 10m;
```

在以上示例中，定义一个叫“one”的记录区，总容量为 10MB，以变量 $binary_remote_addr 作为会话的判断基准（即一个地址一个会话）。

您可能注意到了，在这里使用的是 $binary_remote_addr 而不是 $remote_addr。

$remote_addr 的长度为 7 至 15 bytes，会话信息的长度为 32 或 64 bytes。而 $binary_remote_addr 的长度为 4 bytes，会话信息的长度为 32 bytes。

当记录区的大小为 1MB 的时候，大约可以记录 32 000 个会话信息（一个会话占用 32 bytes）。

13.15.2 limit_conn 指令

语法：limit_conn zone_name max_clients_per_ip

默认值：no

使用环境：http, server, location

该指令用于指定一个会话最大的并发连接数。当超过指定的最大并发连接数时，服务器将返回“Service unavailable”（503）。

示例：

```
limit_zone  one  $binary_remote_addr  10m;

server {
 location /download/ {
 limit_conn  one  1;
}
```

以上示例中，定义一个叫“one”的记录区，总容量为 10MB，以变量 $binary_remote_addr 作为会话的判断基准（即一个地址一个会话）。限制 /download/ 目录下，一个会话只能进行一个连接。简单来说，就是限制 /download/ 目录下，一个 IP 只能发起一个连接，多于一个，一律返回“Service unavailable”（503）状态。

13.16 HTTP Limit Req 模块

该模块允许你对 Session 会话、单个客户端 IP 地址，限制指定单位时间内的并发请求数。你可以在一定程度上减轻对应用服务器的 DOS 恶意攻击。配置示例如代码 13-13 所示：

www.TopSage.com

代码 13-13

```
http {
   limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;

   ...

   server {

      ...

      location /search/ {
         limit_req  zone=one  burst=5;
      }
```

13.16.1 limit_req_zone 指令

语法：limit_req_zone $session_variable zone=name_of_zone:size rate=rate

默认值：none

使用环境：http

该指令用于定义一块内存存储区，用来存储 Session 会话的状态，Session 的变量由指定的变量构成，通常可以设置为存储客户端 IP 的变量$binary_remote_addr。以下是一个示例：

```
limit_req_zone  $binary_remote_addr  zone=one:10m   rate=1r/s;
```

在本例中，为 Session 会话状态分配了一个名为 one 的 10MB 内存存储区，限制了每秒只接受一个 IP 的一次请求（1 Request/Sec）。这里使用变量$binary_remote_addr 代替$remote_addr，可以为每个会话减少 64 字节的内存存储空间，从而使得 1MB 内存可以存储大概 16 000 个会话状态。

速度限制的条件可以设置为“请求数/秒（r/s）”或“请求数/分钟（r/m）”，比率必须为整型数，如果你要设置少于每秒 1 个请求数的值，例如“1 个请求/2 秒”，你应该设置为“30r/m”。

13.16.2 limit_req 指令

语法：limit_req zone=zone burst=burst [nodelay]

默认值：none

使用环境：http, server, location

该指令用于指定使用的内存存储区（zone）名称，以及最大的突发请求数（burse）。如果请

www.TopSage.com

求的速率超过 limit_req_zone 指令中设置的速率，这些请求将被延迟处理，以便能够按照 limit_req_zone 指令中设定的速度处理请求。多余的请求将被延迟处理，直到这些请求的数量不超过 burst 参数中规定的数量。在这种情况下，请求将获得服务不可用信息："Service unavailable"（503 错误代码）。Burst 的值默认为 0。

示例如下：

```
limit_req_zone $binary_remote_addr zone=one:10m  rate=1r/s;

  server {
    location /search/ {
      limit_req  zone=one  burst=5;
    }
```

在该示例中，允许一个用户平均每秒不超过 1 个请求，同时处理的查询数量最多不超过 5 个。如果查询数小于 burst 值，延迟不是必须的，你可以使用 nodelay 参数设置为非延迟：

```
limit_req  zone=one  burst=5  nodelay;
```

13.17　HTTP Log 模块

该指令用于控制 Nginx 的日志格式。示例如下：

```
log_format gzip '$remote_addr - $remote_user [$time_local] '
                '"$request" $status $bytes_sent '
                '"$http_referer" "$http_user_agent" "$gzip_ratio"';

access_log  /spool/logs/nginx-access.log  gzip  buffer=32k;
```

13.17.1　access_log 指令

语法：access_log path [format [buffer=size | off]]

默认值：access_log log/access.log combined

使用环境：http, server, location

该指令用于设置日志文件的路径、格式和缓冲区大小。使用"off"作为唯一参数，将不记录日志文件。如果没有指定日志格式，将默认采用"combined"格式。缓冲区的大小必须小于写入磁盘文件的原子记录大小。这个大小对于 FreeBSD 3.0-6.0 系统无限制。

从 Nginx 0.7.4 版本开始，日志文件的路径可以包含变量，但是有一些限制：

Nginx 的 Worker 用户必须有权限去创建文件；

缓冲将不会工作；

www.TopSage.com

每个请求到来时，日志文件将被打开，在日志写入后，又将立即关闭日志文件。频繁使用的文件描述符将被保存到 open_log_file_cache 指令设置的缓冲区中，关于日志的轮换，必须随着时间的推移被记住（在 open_log_file_cache 指令的参数中设置），日志仍然继续记录在旧的文件中。

Nginx 支持为每个 location 环境设置强大的日志格式。相同的访问可以被同时记录在一个或多个日志文件中。

13.17.2 log_format 指令

语法：log_format name format [format ...]

默认值：log_format combined "..."

使用环境：http, server

该指令用来描述日志格式。在日志的格式中，可以使用 Nginx 的大多数通用变量，以及一些仅在写日志时存在的变量，如下：

$body_bytes_sent——减去应答头之后的传输给客户端的字节数。

$bytes_sent——传输给客户端的字节数。

$connection——连接数。

$msec——正在写日志的当前时间（精确到微秒级，即百万分之一秒）。

$pipe——如果请求是 pipelined 管道。

$request_length——请求的主体（body）长度。

$request_time——请求在 Nginx 开始处理之前所消耗的时间（包括 TCP 连接时间、客户端的 HTTP 发送时间等），单位为毫秒（在 Nginx 0.5.19 版本之前，单位为秒）。

$status——应答状态。

$time_local——写入日志的本地服务器时间。

$http_x_forwarded_for——上级反向代理服务器中通常用于记录用户真实 IP 的 X-Forward 信息。

13.17.3 log_format_combined 指令

Nginx 有一个名为“combined”的预定义（内部已经定义的）指令，相当于以下设置：

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';
```

www.TopSage.com

13.17.4 open_log_file_cache 指令

语法：open_log_file_cache max=N [inactive=time] [min_uses=N] [valid=time] | off

默认值：open_log_file_cache off

使用环境：http server location

该指令用于为带有变量的日志文件路径所频繁使用的文件描述符设置缓存。指令选项如下：

max——缓存中可以存放的最大文件描述符。缓存满之后，根据 LRU 算法删除最早缓存，并且最近没有被使用的文件描述符。

inactive——设置一个时间，在该时间范围内没有被使用的文件描述符将被移除。默认时间为 10 秒。

min_uses——设置一个文件描述符在 inactive 参数规定的单位时间范围内，最少使用的次数。满足最少使用次数的文件描述符才会被放入缓存中。默认的最少使用次数为 1 次。

valid——设置检查同名文件是否存在的时间周期，默认为 60 秒。

off——禁用缓存。

13.18 HTTP Map 模块

该模块允许你去分类，或者映射一组值到一组不同的值，并将这些值存储在一个变量中。示例如代码 13-14 所示：

代码 13-14

```
map $http_host $name {
 hostnames;

 default        0;

 example.com    1;
 *.example.com  1;
 test.com       2;
 *.test.com     2;
 .site.com      3;
}
```

我们可以使用本模块的映射，来代替写许多的 server/location 指令或 redirect 重定向规则，如代码 13-15 所示：

www.TopSage.com

大家网 TopSage.com

代码 13-15

```
map $uri $new {
  default       http://www.domain.com/home/;

  /aa           http://aa.domain.com/;
  /bb           http://bb.domain.com/;
  /john         http://my.domain.com/users/john/;
}

server {
  server_name  www.domain.com;
  rewrite ^    $new  redirect;
}
```

13.18.1 map 指令

语法：map $var1 $var2 { ... }

默认值： none

使用环境：http

map 指令定义了被用来设置变量的映射表，该指令有 3 个特殊的参数：

default——指定无匹配内容时的默认值；

hostnames——支持近似域名查找。示例如下：

```
*.example.com 1;
```

另外，以“.”开头的字符串，将用来精确匹配以该字符串结尾的主机名（域名）。例如，以下两台记录：

```
example.com    1;
*.example.com 1;
```

可以用一条记录来代替：

```
.example.com  1;
```

include——包含一个含有映射信息的文件。它可以包含多个文件。

13.18.2 map_hash_max_size 指令

语法：map_hash_max_size number

默认值：map_hash_max_size 2048

www.TopSage.com

使用环境：http

该指令用于设置保存变量映射关系的哈希表的最大值。

13.18.3 map_hash_bucket_size 指令

语法：map_hash_bucket_size n

默认值：map_hash_bucket_size 32/64/128

使用环境：http

该指令用于设置一张哈希表中映射变量的最大值。默认值取决于处理器的缓存大小。

13.19 HTTP Memcached 模块

Memcached 是一个高性能的分布式内存对象缓存系统，用于动态 Web 应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提供动态、数据库驱动网站的速度。Memcached 基于一个存储键/值对的 hashmap。其守护进程（daemon）是用 C 写的，但是客户端可以用任何语言来编写，并通过 Memcached 协议与守护进程通信，且它并不提供冗余（例如，复制其 hashmap 条目）；当某个服务器停止运行或崩溃了，所有存放在该服务器上的键/值对都将丢失。

Memcached（官方网站：*http://www.memcached.org/*）由 Danga Interactive 开发，用于提升 LiveJournal.com 访问速度。LJ 每秒动态页面访问量几千次，用户 700 万。Memcached 将数据库负载大幅降低，以便更好地分配资源，更快地访问。

Memcached 服务器端的编译安装如代码 13-16 所示：

代码 13-16

```
wget http://www.monkey.org/~provos/libevent-1.4.13-stable.tar.gz
tar zxvf libevent-1.4.13-stable.tar.gz
cd libevent-1.4.13-stable/
./configure --prefix=/usr
make && make install
cd ../

wget http://memcached.googlecode.com/files/memcached-1.4.4.tar.gz
tar zxvf memcached-1.4.4.tar.gz
cd memcached-1.4.4/
./configure --with-libevent=/usr
make && make install
cd ../
```


www.TopSage.com

启动 Memcached 服务器端：

```
memcached -d -m 2048 -l 192.168.1.2 -p 11211
```

Nginx 中的 Memcached 模块，相当于 Memcached 的客户端软件，使用本模块可以完成一些简单的缓存任务。Nginx 官方计划在不久的将来继续完善这一模块。

配置示例如代码 13-17 所示：

代码 13-17

```
server {
  location / {
    set $memcached_key $uri;
    memcached_pass     name:11211;
    default_type       text/html;
    error_page         404 = /fallback;
  }

  location = /fallback {
    proxy_pass backend;
  }
}
```

或者在 Nginx 0.7.x 版本中使用如代码 13-18 所示的内容：

代码 13-18

```
server {
  location / {
    set $memcached_key $uri;
    memcached_pass     name:11211;
    default_type       text/html;
    error_page         404 @fallback;
  }

  location @fallback {
    proxy_pass backend;
  }
}
```

接下来，我们结合 PHP，使用 Memcached 模块做缓存。在代码 13-19 所示示例中，如果用户访问以“/cache/”开始的 URI，则读取以 URI 为 key 的 Memcached 缓存内存，显示给用户。如果 key 不存在，则重定向到“/write_memcached.php”程序文件，该 PHP 程序会往 Memcached 内写入以 URI 为 key 的缓存数据，缓存时间为 120 秒。

代码 13-19

```
location /cache/
{
   memcached_buffer_size 10240;
```


www.TopSage.com




```
    set $memcached_key $request_uri;
    memcached_pass 127.0.0.1:11211;
    error_page 404 =200 /write_memcached.php;
}

location ~ .*\.(php|php5)?$
{
    #fastcgi_pass  unix:/tmp/php-cgi.sock;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
    fastcgi_param  MEMCACHED_KEY     $memcached_key;
}
```

write_memcached.php 文件的内容如代码 13-20 所示：

代码 13-20

```
<?php
$key = $_SERVER['MEMCACHED_KEY'];
$value = "<html><head><title>标题</title></head><body>".date("Y-m-d H:i:s")."
<BR><BR></body></html>";

$memcache = new Memcache;
$memcache->connect('127.0.0.1', 11211);
$memcache->set($key, $value, false, 120); //缓存时间为 120 秒
$memcache->close();

echo $value;

//另输出一段信息，用来检验 Memcached 缓存是否有效。如果 Memcached 无缓存、缓存过期，则会显示下行
信息，否则不会。
echo "这是动态 PHP 页面";
?>
```

13.19.1 memcached_pass 指令

语法：memcached_pass [name:port]

默认值：none

使用环境：http, server, location

该指令用于设置 Memcached 服务器的地址和端口，Memcached 的 key 为“/uri?args”。

从 Nginx 0.5.9 版本开始，Memcached 的 key 存储在$memcached_key 变量中。

www.TopSage.com

13.19.2 memcached_connect_timeout 指令

语法：memcached_connect_timeout [time]

默认值：60000

使用环境：http, server, location

该指令用于设置连接 Memcached 服务器的超时时间，单位为毫秒。

13.19.3 memcached_read_timeout 指令

语法：memcached_read_timeout [time]

默认值：60000

使用环境：http, server, location

该指令用于设置从 Memcached 服务器读取数据的超时时间，单位为毫秒。

13.19.4 memcached_send_timeout 指令

语法：memcached_send_timeout [time]

默认值：60000

使用环境：http, server, location

该指令用于设置向 Memcached 服务器发送数据的超时时间，单位为毫秒。

13.19.5 memcached_buffer_size 指令

语法：memcached_buffer_size [size]

默认值：see getpagesize（2）

使用环境：http, server, location

该指令用于设置接收、发送数据的缓冲区大小，单位为字节。

13.19.6 memcached_next_upstream 指令

语法：memcached_next_upstream [error | timeout | invalid_response | not_found | off]

www.TopSage.com

默认值：error timeout

使用环境：http, server, location

该指令用于指定在哪种错误条件下，将请求转发到 upstream 负载均衡服务器池中的另一台服务器。仅适用于当 upstream 中的 memcached_pass 指令拥有两个或两个以上后端服务器的情况。

13.19.7　HTTP Memcached 模块中的变量

```
$memcached_key
```

该变量的内容为 Memcached 的 key 值。通常情况下，以当前请求的 URI 作为 Memcached 的 key，可以设置变量为：

```
set $memcached_key $uri;
```

13.19.8　第三方的 Memcached 模块

除了 Nginx 官方的 Memcached 模块之外，还有一个第三方的 Memcached 模块，几乎支持 Memcached 所有的功能。但是，考虑到第三方模块通常缺乏稳定性与兼容性，这里就不再详细介绍，感兴趣的朋友可以访问：http://wiki.nginx.org/NginxHttpMemcModule，获取更多的信息。

13.20　HTTP Proxy 模块

该模块用于转发请求到其他的服务器。Keep-alive 是指在 HTTP/1.1 协议中，同一个连接中发出和接收多次 HTTP 请求，节省了创建 TCP 连接过程的时间开销。而 HTTP/1.0 协议不具备 keep-alive 请求的能力。因此，在 HTTP/1.0 协议中，每一个到后端的请求都会创建一个连接，传输完成后会删除这个连接。Nginx 采用 HTTP/1.1 协议与浏览器进行通信，采用 HTTP/1.0 协议与后端服务器进行通信。

示例如下：

```
location / {
  proxy_pass        http://localhost:8000;
  proxy_set_header  X-Real-IP  $remote_addr;
}
```

注意：当使用 HTTP 代理模块时（或者当使用 FastCGI 时），整个客户端请求在传递给后端服务器之前，将被 Nginx 缓存，数据传输的进度测量将不准确。

www.TopSage.com

大家网 TopSage.com

13.20.1 proxy_buffer_size 指令

语法：proxy_buffer_size the_size

默认值：proxy_buffer_size 4k/8k

使用环境：http, server, location

该指令用于设置从被代理服务器获取的第一部分应答信息的缓冲区大小，这个缓存区间会保存用户的头信息以供 Nginx 进行规则处理，一般只要能保存下头信息即可。但是，也可以将该值设置得较小。

13.20.2 proxy_buffering 指令

语法：proxy_buffering on|off

默认值：proxy_buffering on

使用环境：http, server, location

该指令用于开启对被代理的服务器的应答缓冲。

如果缓冲开启，Nginx 会假设被代理服务器的应答尽可能地快，并将其保存到缓冲区。可以使用指令 proxy_buffer_size 和 proxy_buffers 来配置缓冲区信息。

如果应答内容无法完全放在内存中，那么应答内容的一部分将被写入在磁盘。

如果缓存关闭，从后端服务器接收到的应答内容，将被立即同步传送到客户端。

Nginx 不会尝试统计被代理服务器的全部应答数、数据的最大值，Nginx 能够接受 proxy_buffer_size 指令设置的值。

对于长轮询的计算机信息传输应用，应该尽可能地设置 proxy_buffering 为 off，否则异步地应答将被缓存，计算机信息传输将不工作。

13.20.3 proxy_buffers 指令

语法：proxy_buffers the_number is_size;

默认值：proxy_buffers 8 4k/8k;

使用环境：http, server, location

www.TopSage.com

该指令用于设置从被代理服务器读取应答信息的缓冲区数目和大小。通常情况下，一个缓冲区的大小相当于网页的大小。该指令的默认缓冲区大小取决于操作系统平台，不是 4k 就是 8k。

假设你的网页平均大小都在 32k 以下，则可以设置：

```
proxy_buffers          4 32k;
```

13.20.4 proxy_busy_buffers_size 指令

语法：proxy_busy_buffers_size size;

默认值：proxy_busy_buffers_size ["#proxy buffer size"] * 2;

使用环境：http, server, location, if

系统很忙的时候可以申请更大的 proxy_buffers 缓冲区，官方推荐为 proxy_buffers 缓冲区的大小*2，例如：

```
proxy_busy_buffers_size 64k;
```

13.20.5 proxy_cache 相关指令集

proxy_cache 相关指令集，包含 proxy_cache、proxy_cache_path、proxy_cache_methods、proxy_cache_min_uses、proxy_cache_valid、proxy_cache_use_stale 等指令，在前面第 9 章中已经有详细说明，这里就不再重复介绍。

13.20.6 proxy_connect_timeout 指令

语法：proxy_connect_timeout timeout_in_seconds

使用环境：http, server, location

该指令用于设置跟后端服务器连接的超时时间。该时间不是服务器返回页面的时间，而是发起握手等候响应的超时时间。

13.20.7 proxy_headers_hash_bucket_size 指令

语法：proxy_headers_hash_bucket_size size;

默认值：proxy_headers_hash_bucket_size 64;

使用环境：http, server, location, if

www.TopSage.com

该指令用于设置哈希表的存储桶数量。

13.20.8 proxy_headers_hash_max_size 指令

语法：proxy_headers_hash_max_size size;

默认值：proxy_headers_hash_max_size 512;

使用环境：http, server, location, if

该指令用于设置哈希表的最大值。

13.20.9 proxy_hide_header 指令

语法：proxy_hide_header the_header

默认值：http, server, location

Nginx 不会传输从被代理服务器应答内容获取的"Date"、"Server"、"X-Pad"和"X-Accel-..."等 Header 行。proxy_hide_header 指令允许隐藏一些额外的 Header 行。除了必须被传递的 Header 行，其他 Header 行都可以使用 proxy_pass_header 指令来隐藏。例如你可以按照以下方法隐藏 MS-OfficeWebserver 和 AspNet-Version：

```
location / {
  proxy_hide_header X-AspNet-Version;
  proxy_hide_header MicrosoftOfficeWebServer;
}
```

13.20.10 proxy_ignore_client_abort 指令

语法：proxy_ignore_client_abort [onloff]

默认值：proxy_ignore_client_abort off

使用环境：http, server, location

如果客户端自身终止请求，防止中断代理请求。

13.20.11 proxy_ignore_headers 指令

语法：proxy_ignore_headers name [name ...]

默认值：none

大家网 TopSage.com

使用环境：http, server, location

Nginx 0.7.54 以上版本开始支持该指令。该指令可以禁止处理从代理服务器返回的 Header 行。它可以指定字符串，例如："X-Accel-Redirect"、"X-Accel-Expires"、"Expires" 或 "Cache-Control"。

13.20.12 proxy_intercept_errors 指令

语法：proxy_intercept_errors [on|off]

默认值：proxy_intercept_errors off

使用环境：http, server, location

该指令判断 Nginx 是否会拦截 HTTP 状态码为 400 及以上代码的应答。

默认情况下，来自被代理服务器的所有应答将照常发送。

如果设定 proxy_intercept_errors on，Nginx 将会拦截 error_page 指令明确指定的错误状态码。如果来自被代理服务器的应答状态码不匹配一个 error_page 指令，应答将被照常发送。

13.20.13 proxy_max_temp_file_size 指令

语法：proxy_max_temp_file_size size;

默认值：proxy_max_temp_file_size 1G;

使用环境：http, server, location, if

该指令用于设置当网页内容大于代理内存缓冲区的时候，临时文件大小的最大值。如果文件大于这个值，它将从 upstream 服务器同步地传递请求，而不是缓冲到磁盘。

13.20.14 proxy_method 指令

语法：proxy_method [method]

默认值：None

使用环境：http, server, location

该指令允许代理额外的 HTTP 方法。HTTP 协议除了常用的 GET、POST 方法外，还有一些其他的方法，例如 Squid 缓存服务器使用的自定义 PURGE 方法。

注意：当使用本指令时，Nginx 只允许单个 HTTP 方法参数，所以目前还不能代理类似

www.TopSage.com

Subversion 版本控制服务的请求。

proxy_method 指令使用示例：

```
proxy_method PROPFIND;
```

13.20.15 proxy_next_upstream 指令

语法：proxy_next_upstream

[error|timeout|invalid_header|http_500|http_502|http_503|http_504| http_404|off]

默认值：proxy_next_upstream error timeout

使用环境：http, server, location

该指令用于设置当在哪种情况下，将请求转发到下一台服务器。在 upstream 负载均衡代理服务器池中，假设后端的一台服务器无法访问或返回指定错误响应代码时，可以使用该指令将请求转发到池中的下一台服务器。

可以设置的错误响应参数如下：

error——如果连接服务器时、发送请求时、读取应答信息时发送错误。

timeout——如果连接服务器时、传递请求时、读取后端服务器应答信息时超时。

invalid_header——后端服务器返回一个空的或错误的应答时。

http_500——后端服务器返回 500 应答状态码时。500 状态码一般是 Nginx 配置、FastCGI 程序出错导致。

http_502——后端服务器返回 500 应答状态码时。502 状态码一般是后端服务器程序执行超时、无法访问导致。

http_503——后端服务器返回 503 应答状态码时。

http_504——后端服务器返回 504 应答状态码时。

http_404——后端服务器返回 500 应答状态码时。404 状态码一般是文件不存在导致。

off——禁止将请求转发到下一台后端服务器。

当没有任何数据传送到客户端时，将请求转发到下一台后端服务器才会生效。确切地说，如果在传送应答内容给客户端的过程中出错或超时，将不会尝试重新请求一个不同的后端服务器。

www.TopSage.com

13.20.16 proxy_pass 指令

语法：proxy_pass URL

默认值：no

使用环境：location, if in location

该指令用于设置被代理服务器端口或套接字，以及 URI。

可以使用域名（IP 地址）和端口来指定被代理的服务器，例如：

```
proxy_pass http://localhost:8000/uri;
```

也可以使用 Unix 套接字来指定被代理的服务器，例如：

```
proxy_pass unix:/path/to/backend.socket;
```

请求转发给后端服务器的 URI 部分，取自 Nginx 代理服务器的 location 后面的 URI 路径部分。例如在 Nginx 代理服务器的以下配置中，访问/name/test/test.html 将反向代理访问 http://192.168.1.2/test/test.html 的内容：

```
location /name/ {
proxy_pass  http://192.168.1.2;
}
```

但是，对于以上的规则有以下两个例外：

（1）当以正则表达式的方式指定 location 时。

（2）在 location 中使用 rewrite 规则改变 URI 时，使用这个配置可以能够更精确地处理请求（break）：

```
location  /name/ {
 rewrite    /name/([^/] +)  /users?name=$1  break;
 proxy_pass   http://127.0.0.1;
}
```

在这些 URI 的传递案例中没有使用映射。

此外，须指明，以下 URI 将使用同样的格式转发。在工作过程中：

两个或多个连续的斜线将被转换成一个斜线："//" -- "/"；

引用的当前目录会被删除："/./" -- "/"；

引用的主目录会被删除："/dir/../" -- "/"。

如果一个服务器上要传送未经处理的格式，那么 proxy_pass 的参数则要使用不含 URI 的 URL，例如：

www.TopSage.com

大家网 TopSage.com

```
location /some/path/ {
 proxy_pass  http://127.0.0.1;
}
```

在 proxy_pass 指令中使用变量是一个特例，也可以使用 rewrite 和 proxy_pass 的组合：

```
location / {
 rewrite ^(.*)$ /VirtualHostBase/https/$server_name:443/some/path/
  VirtualHostRoot/$1 break;
 proxy_pass  http://127.0.0.1:8080;
}
```

13.20.17 proxy_pass_header 指令

语法：proxy_pass_header the_name

使用环境：http, server, location

该指令允许转发为应答转发 Header 行。

```
location / {
 proxy_pass_header X-Accel-Redirect;
}
```

13.20.18 proxy_pass_request_body 指令

语法：proxy_pass_request_body [on | off] ;

默认值：proxy_pass_request_body on;

使用环境：http, server, location

13.20.19 proxy_pass_request_headers 指令

语法：proxy_pass_request_headers [on | off] ;

默认值：proxy_pass_request_headers on;

使用环境：http, server, location

13.20.20 proxy_redirect 指令

语法：proxy_redirect [default|off|redirect replacement]

默认值：proxy_redirect default

使用环境：http, server, location

该指令用于更改被代理服务器的应答 Header 头中的"Location"和"Refresh"。

我们假设被代理服务器返回的行是：Location: *http://localhost:8000/two/some/uri/*，那么，以下指令：

```
proxy_redirect http://localhost:8000/two/ http://frontend/one/;
```

会将此行重写为 Location: *http://frontend/one/some/uri/*。

在代替的行中可以不用写服务器的名称，只用写相对路径就可以了，例如：

```
proxy_redirect http://localhost:8000/two/ /;
```

这样，就设置了服务器的基本名称和端口，即使它不是 80 端口。

默认参数“default”，取决于 location 和 proxy_pass 指令的值。

例如以下两个配置的效果是一样的，这里的参数“default”的值，取决于 proxy_pass 指令的参数“*http://upstream:port/two/*”和 location 指令的参数“/one/”：

```
location /one/ {
  proxy_pass       http://upstream:port/two/;
  proxy_redirect   default;
}

location /one/ {
  proxy_pass       http://upstream:port/two/;
  proxy_redirect   http://upstream:port/two/   /one/;
}
```

在代替的行中，可以使用一些 Nginx 变量，例如：

```
proxy_redirect   http://localhost:8000/    http://$host:$server_port/;
```

该指令有时候可以重复：

```
proxy_redirect   default;
proxy_redirect   http://localhost:8000/    /;
proxy_redirect   http://www.example.com/   /;
```

参数 off 在这个使用环境（例如 location /one/ {......}）中禁止所有的 proxy_redirect 指令：

```
proxy_redirect   off;
proxy_redirect   default;
proxy_redirect   http://localhost:8000/    /;
proxy_redirect   http://www.example.com/   /;
```

以下这条语句可以帮助被代理服务器为相对重定向添加主机名：

```
proxy_redirect   /   /;
```


www.TopSage.com

13.20.21 proxy_read_timeout 指令

语法：proxy_read_timeout the_time

默认值：proxy_read_timeout 60

使用环境：http, server, location

该指令用于设置从后端被代理服务器读取应答内容的超时时间。它决定 Nginx 等待多长时间来取得一个请求的应答。超时时间是指完成 TCP 两次握手，到 TCP 状态为 ESTABLISHED 时（数据正在传输）的时间，而不是整个时间。

注意不要将此时间设置得太低，如果被代理服务器超过此时间，仍然没有传输数据，Nginx 将关闭该次连接。

13.20.22 proxy_redirect_errors 指令

目前，已经不赞成使用该指令。请使用 proxy_intercept_errors 指令。

13.20.23 proxy_send_lowat 指令

语法：proxy_send_lowat [on | off]

默认值：proxy_send_lowat off;

使用环境：http, server, location, if

该指令用于设置是否使用 SO_SNDLOWAT，该指令只可以在 FreeBSD 操作系统中使用。

13.20.24 proxy_send_timeout 指令

语法：proxy_send_timeout time_in_seconds

默认值：proxy_send_timeout 60

使用环境：http, server, location

该指令用于指定代理服务器转发请求的超时时间。超时时间是指完成 TCP 两次握手，到 TCP 状态为 ESTABLISHED 时（数据正在传输）的时间，而不是整个时间。如果代理服务器超过此时间，仍然没有转发数据到后端服务器，Nginx 将关闭该次连接。

www.TopSage.com

大家网 TopSage.com

13.20.25 proxy_set_body 指令

语法：proxy_set_body [on | off]

默认值：proxy_set_body off;

使用环境：http, server, location, if

从 Nginx 0.3.10 版本开始，可以使用该指令。

13.20.26 proxy_set_header 指令

语法：proxy_set_header header value

默认值：Host and Connection

使用环境：http, server, location

该指令允许重新定义或添加 Header 行到转发给被代理服务器的请求信息中，它的值可以是文本，也可以是变量，或者是文本和变量的组合。

当 proxy_set_header 指令没有对指定的 Header 行进行定义时，将从客户端请求的 Header 行中继承这些行的信息。默认情况下，只有以下两行可以被重新定义：

```
proxy_set_header Host $proxy_host;
proxy_set_header Connection Close;
```

未修改的 Header 头“Host”能够按照如下方式发送：

```
proxy_set_header Host $http_host;
```

然而，如果客户端请求中没有这个 Header 头，将没有“Host”行的数据发送给被代理服务器。在这种情况下，最好使用$host 变量，它的值相当于服务器的主机名（如果使用域名访问，则该值为域名；如果使用 IP 访问，则该值为 IP）：

```
proxy_set_header Host $host;
```

此外，可以将主机名和被代理服务器的端口一起传递：

```
proxy_set_header Host $host:$proxy_port;
```

如果设置指定 Header 行的值为空，该 Header 行将不会被发送到 upstream。例如，以下示例可以用来在 upstream 中禁止后端服务器使用 gzip 压缩：

```
proxy_set_header Accept-Encoding "";
```


www.TopSage.com

13.20.27 proxy_store 指令

语法：proxy_store [on | off | path]

默认值：proxy_store off

使用环境：http, server, location

该指令可以设置哪些从后端服务器传送过来的文件被 Nginx 存储。参数"on"保持文件与 alias 或 root 指令设置的目录一致，参数"off"不存储文件。此外，路径名称中可以使用变量，例如：

```
proxy_store   /data/www$original_uri;
```

文件的最后修改时间将被在应答数据的"Last-Modified" Header 头中设定。为了文件的安全，可以使用 proxy_temp_path 指令设置一个临时目录。

该指令可以为不经常修改的文件建立一份本地镜像，从而减轻后端被代理服务器的压力。使用示例如代码 13-21 所示：

代码 13-21

```
location /images/ {
  root              /data/www;
  error_page         404 = /fetch$uri;
}

location /fetch {
  internal;
  proxy_pass         http://backend;
  proxy_store        on;
  proxy_store_access   user:rw  group:rw  all:r;
  proxy_temp_path     /data/temp;
  alias             /data/www;
}
```

或者代码 13-22 中的内容：

代码 13-22

```
location /images/ {
  root              /data/www;
  error_page         404 = @fetch;
}

location @fetch {
  internal;

  proxy_pass         http://backend;
  proxy_store        on;
```


www.TopSage.com

```
  proxy_store_access   user:rw  group:rw  all:r;
  proxy_temp_path     /data/temp;

  root                /data/www;
}
```

注意：proxy_store 不是缓存，存储的文件不会自动过期，它相当于一个镜像。

13.20.28 proxy_store_access 指令

语法：proxy_store_access users:permissions [users:permission ...]

默认值：proxy_store_access user:rw

使用环境：http, server, location

该指令用于指定创建文件和目录的权限，例如：

```
proxy_store_access  user:rw  group:rw  all:r;
```

如果指定的用户组是正确的，那么，不需要指定用户的权限，例如：

```
proxy_store_access  group:rw  all:r;
```

13.20.29 proxy_temp_file_write_size 指令

语法：proxy_temp_file_write_size size;

默认值：proxy_temp_file_write_size ["#proxy buffer size"] * 2;

使用环境：http, server, location, if

设置写入 proxy_temp_path 临时目录的数据大小。它可以防止一个工作进程阻塞太长时间。

13.20.30 proxy_temp_path 指令

语法：proxy_temp_path dir-path [level1 [level2 [level3] ;

默认值：$NGX_PREFIX/proxy_temp controlled by --http-proxy-temp-path at ./configure stage

使用环境：http, server, location

该指令工作方式类似 client_body_temp_path，用于指定一个本地目录来缓冲较大的代理请求。

www.TopSage.com

大家网 TopSage.com

13.20.31 proxy_upstream_fail_timeout 指令

从 Nginx 0.5.0 版本开始，不赞成使用该指令。请使用 upstream 模块中的 fail_timeout 参数来代替它。

13.20.32 proxy_upstream_max_fails 指令

从 Nginx 0.5.0 版本开始，不赞成使用该指令。请使用 upstream 模块中的 max_fails 参数来代替它。

13.20.33 HTTP Proxy 模块的变量

HTTP Proxy 模块包含一些内置变量，可以用于 proxy_set_header 指令：

`$proxy_host`

被代理服务器的主机名和端口。

`$proxy_port`

被代理服务器的端口。

`$proxy_add_x_forwarded_for`

包含以半角逗号分割的客户端请求头“X-Forwarded-For”和$remote_addr 的内容。如果没有请求头“X-Forwarded-For”，该变量的值等于$remote_addr 的值。

13.21 HTTP Rewrite 模块

该模块及相关指令已经在第 7 章中有详细说明，这里就不再重复介绍。

13.22 HTTP SSI 模块

SSI（Server Side Include），通常称为服务器端嵌入。

SSI 的工作原理：将内容发送到浏览器之前，可以使用“服务器端包含 （SSI）”指令将文本、图形或应用程序信息包含到网页中。例如，可以使用 SSI 包含时间/日期戳、版权声明或供客户填写并返回的表单。对于在多个文件中重复出现的文本或图形，使用包含文件是一种简便的方法。将内容存入一个包含文件中即可，而不必将内容输入所有文件。通过一个非常简单的语句

www.TopSage.com

即可调用包含文件，此语句指示 Web 服务器将内容插入适当网页。而且，使用包含文件时，对内容的所有更改只需在一个地方就能完成。

因为包含 SSI 指令的文件要求特殊处理，所以必须为所有 SSI 文件赋予 SSI 文件扩展名。默认扩展名是“.shtml”。例如以下的网址，则使用了 SSI：

```
http://finance.sina.com.cn/g/20091222/16047138654.shtml
```

Nginx 支持 SSI，但是，目前功能并不是十分完善。

13.22.1 ssi 指令

语法：ssi [on | off]

默认值：ssi off

使用环境：http, server, location, if in location

该指令用于开启或关闭 SSI 处理。

13.22.2 ssi_silent_errors 指令

语法：ssi_silent_errors [on|off]

默认值：ssi_silent_errors off

使用环境：http, server, location

该指令设置为 off 时，当处理 SSI 时发生错误，不输出“[an error occurred while processing the directive]”错误信息。

13.22.3 ssi_types 指令

语法：ssi_types mime-type [mime-type ...]

默认值：ssi_types text/html

使用环境：http, server, location

允许 SSI 处理其他 MIME 类型。SSI 默认可以处理“text/html”类型。

Enables SSI processing for MIME-types in addition to "text/html" types.

www.TopSage.com

13.22.4 ssi_value_length 指令

语法：ssi_value_length length

默认值：ssi_value_length 256

使用环境：http, server, location

定义 SSI 中允许使用的参数长度。

13.22.5 SSI 命令

指令格式如下：

```
<!--# command parameter1=value parameter2=value... -->
```

支持的 SSI 命令列表如下：

1. block

该命令允许创建一个块，在块中可以使用 SSI 命令：

name——块的名称。示例：

```
<!--# block name="one" -->
the silencer
<!--# endblock -->
```

2. config

为 SSI 配置一些参数。

- errmsg——SSI 处理过程中的错误行，默认字符串为：[an error occurred while processing the directive]。
- timefmt——时间字符串的格式，strftime(3)函数使用，默认为：

```
"%A, %d-%b-%Y %H:%M:%S %Z"
```

时间中的秒可以使用“%s”。

3. echo

打印一个变量。

- var——变量的名称。
- default——如果变量为空，则显示这个字符串，默认值为“none”。示例：

www.TopSage.com

大家网 TopSage.com

```
<!--# echo var="name" default="no" -->
```

与以下写法的功能相似：

```
<!--# if expr="$name" --><!--# echo var="name" --><!--# else -->no<!--# endif -->
```

4. if/elif/else/endif

使用方法：

```
<!--# if expr="..." -->
...
<!--# elif expr="..." -->
...
<!--# else -->
...
<!--# endif -->
```

以上示例只允许一层嵌套，不允许多层嵌套。

- expr——匹配一个表达式，可以是变量：

```
<!--# if expr="$name" -->
```

字符串比较：

```
<!--# if expr="$name = text" -->
<!--# if expr="$name != text" -->
```

正则匹配：

```
<!--# if expr="$name = /text/" -->
<!--# if expr="$name != /text/" -->
```

如果使用变量，则用它们的值代替。

5. include

包含一个其他来源的文档。

- file——包含一个文件，例如：

```
<!--# include file="footer.html" -->
```

- virtual——包含一个请求，例如：

```
<!--# include virtual="/remote/body.php?argument=value" -->
```

多个请求将并行执行，如果要按顺序执行，请使用“wait”选项。

- stub——如果请求为空或返回一个错误后使用的默认块。

```
<!--# block name="one" --> <!--# endblock -->
<!--# include virtual="/remote/body.php?argument=value" stub="one" -->
```

- wait——当设置为 yes 时，在当前的请求完成之前，剩余的 SSI 不会判定，示例：

www.TopSage.com

```
<!--# include virtual="/remote/body.php?argument=value" wait="yes" -->
```

6. set

设置一个变量。

- var——变量。
- value——包含变量名变量的值，它们将被匹配。

13.22.6 SSI 变量

SSI 变量代码格式如下：

```
$date_local
```

本地时区的当前时间，选项 "timefmt"控制格式。

```
$date_gmt
```

当前的格林尼治时间，选项 "timefmt"控制格式。

13.23 HTTP Userid 模块

该模块相当于 Apache 的 mod_uid 模块，主要用于做客户端的身份标识。它主要使用$uid_got 和$uid_set 变量。

配置示例：

```
userid         on;
userid_name     uid;
userid_domain   example.com;
userid_path     /;
userid_expires  365d;
userid_p3p      'policyref="/w3c/p3p.xml", CP="CUR ADM OUR NOR STA NID"';
```

13.23.1 userid 指令

语法：userid [on|v1|log|off]

默认值：userid off

使用环境：http, server, location

该指令允许或禁止发布 cookie 及记录请求的 cookie：

www.TopSage.com

on——允许使用版本 2 的 cookie 及记录它们。

v1——允许使用版本 1 的 cookie 及记录它们。

log——不发送 cookie，但是写下 cookie 来记录。

off——禁止发送及写 cookie。

13.23.2　userid_domain 指令

语法：userid_domain [name | none]

默认值：userid_domain none

使用环境：　http, server, location

指定 cookie 的域名。参数“none”不为 cookie 指定域名。

13.23.3　userid_expires 指令

语法：userid_expires [time | max]

默认值：none

使用环境：http, server, location

设置 cookie 的过期时间。

参数设置并发出浏览器 cookie 的过期时间。参数值“max”指定过期时间为“2037 年 12 月 31 日 23:55:55 GMT”，这是一些老的浏览器能够识别的最大时间。

13.23.4　userid_name 指令

语法：userid_name name

默认值：userid_name uid

使用环境：http, server, location

该指令用于指定 cookie 的名称。

13.23.5　userid_p3p 指令

语法：userid_p3p line

 www.TopSage.com

大家网 TopSage.com

默认值：none

使用环境：http, server, location

Header 头“P3P”的值，将和 cookie 一起传递。

13.23.6 userid_path 指令

语法：userid_path path

默认值：userid_path /

使用环境：http, server, location

设置 cookie 的路径。

13.23.7 userid_service 指令

语法：userid_service number

默认值：userid_service address

使用环境：http, server, location

该指令用于设置发出 cookie 的服务器 IP 地址。如果不设置。版本 1 的 cookie 设置为 0，版本 2 的 cookie 设置为服务器的 IP。

www.TopSage.com

第14章

Nginx 的其他 HTTP 模块

本章介绍的 HTTP 模块不会在编译 Nginx 时自动编译进来，除非使用“./configure --with-模块名”命令指定编译这些模块到 Nginx 中。

在本章的指令介绍中，指令的“使用环境”是指该指令可以在 Nginx 配置文件中使用的位置，例如使用环境为“http, server, location”，则表示该指令可以在以下位置使用：

http { ...... }大括号内；server { ...... }大括号内；location { ...... }大括号内。

14.1 HTTP Addition 模块

本模块可以在当前的 location 内容之前或之后添加其他的 location 内容。

它作为一个输出过滤器被执行，主请求的内容和到其他 location 的内容不会被完全缓冲，将仍以流处理的方式发送到客户端。因为当发送 HTTP Header 头信息时，最终响应 Body 主体的长度是未知的，所以这里 Nginx 将不会在 Header 头中提供 Content-Length 头信息，而是始终采用 HTTP Chunked 编码动态地提供 body 内容的长度。进行 Chunked 编码传输的 HTTP 响应会在 Header 头中设置：Transfer-Encoding: chunked，表示 Body 主体将用 Chunked 编码传输内容。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_addition_module
```

大家网 TopSage.com

本模块相关指令的使用示例如下：

```
location / {
  add_before_body   /before_action;
  add_after_body    /after_action;
}
```

限制项如下：

在 Nginx 0.8.17 中，如果当前的 location 为它自身的子请求提供服务，该子请求的内容将不会被添加，例如代码 14-1 示例：

代码 14-1

```
location /foo {
  add_before_body /bar;
}

location /bar {
  add_before_body  /baz;
}
```

访问/foo 不会使用/baz 代替子请求/bar 之前的内容。另外，在 before/after body localtion 中，只能使用字符串，而不能使用变量。以下配置虽然可以在 nginx.conf 配置文件检测时通过，但是不能正常工作。

```
location / {
  set $before_action /before_action;
  add_before_body $before_action;
}
```

14.1.1 add_before_body 指令

语法：add_before_body uri

默认值：no

使用环境：http, server, location

该指令用于在应答主体之前添加 URI 子请求的内容结果。

14.1.2 add_after_body 指令

语法：add_after_body uri

默认值：no

使用环境：http, server, location

www.TopSage.com

该指令用于在应答主体之后添加 URI 子请求的内容结果。

14.1.3 addition_types 指令

语法：addition_types mime-type [mime-type ...]

默认值：text/html

使用环境：http, server, location

该指令（从 Nginx 0.7.9 版本开始）允许 location 处理自定义的 MIME 类型（默认为“text/html”）。

在 Nginx 0.8.17 之前的版本，该指令在源代码中被错误地拼写为“addtion_types”，这个 Bug 在 0.8.17 版本中被修正。

14.2 Embedded Perl 模块

本模块允许在 Nginx 中直接执行 Perl，或者通过 SSI 调用 Perl。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_perl_module
```

另外，您的操作系统中必须安装 Perl 5.6.1 以上版本。

已知问题：

（1）如果 Perl 模块执行长时间的操作，例如 DNS 查询、数据库查询等，运行 Perl 脚本的工作进程将一直处于阻塞状态，因此，内置的 Perl 脚本应该非常简单，执行尽可能快。

（2）Nginx 在通过“kill -HUP <pid>”命令重新加载配置文件时，可能会导致内存泄露。

示例如代码 14-2 所示：

代码 14-2

```
http {
 perl_modules  perl/lib;
 perl_require  hello.pm;

 perl_set  $msie6  '
 sub {
  my $r = shift;
  my $ua = $r->header_in("User-Agent");
  return "" if $ua =~ /Opera/;
  return "1" if $ua =~ / MSIE [6-9] \.\d+/;
  return "";
```

```
  }
 ';

  server {
    location / {
      perl  hello::handler;
    }
  }
}
```

perl/lib/hello.pm 示例如代码 14-3 所示：

代码 14-3

```
package hello;
use nginx;

sub handler {
  my $r = shift;
  $r->send_http_header("text/html");
  return OK if $r->header_only;

  $r->print("hello!\n<br/>");
  $r->rflush;

  if (-f $r->filename or -d _) {
    $r->print($r->uri, " exists!\n");
  }

  return OK;
}

1;
__END__
```

14.2.1 perl 指令

语法：perl module::function | 'sub {...}'

默认值：no

使用环境：location

该指令用来指定数据 location 中须要用到的 Perl 函数。

14.2.2 perl_modules 指令

语法：perl_modules path

www.TopSage.com

大家网 TopSage.com

默认值：no

使用环境：http

该指令用于为 Perl 模块指定额外的路径。从 Nginx 0.6.7 版本开始，该路径与 nginx.conf 配置文件所在目录有关，而不是 Nginx 的 prefix 安装目录。

14.2.3 perl_require 指令

语法：perl_require module

默认值：no

使用环境：http

这里，你可以使用多个 perl_require 指令。

14.2.4 perl_set 指令

语法：perl_set module::function | 'sub {...}'

默认值：no

使用环境：http

该指令用于设置一个名称为变量名的函数体。示例如代码 14-4：

代码 14-4

```
......
http {
 perl_set  $path_md5  '
   use Digest::MD5 qw(md5_hex);
   use File::stat;

   sub {
      my $r = shift;
      my $s = md5_hex($r->uri);
      my $path_md5 = join "", join("/", substr($s, 0, 1), substr($s, 1, 1),
         substr($s, 2)), ".html";
      my $filepath = "/data/www/".$path_md5;
      if(-f $filepath) {
         my $mtime = stat($filepath)->mtime;
         if(time() - $mtime > 1800) {
            return $path_md5.".new";
         }
      }
      return $path_md5;
```

www.TopSage.com

```
    }';

  server {
    listen 80;
    server_name 127.0.0.1;
    index index.html;
    root   /data/www;
    location /images/ {
       try_files /$path_md5 @fastcgi;
    }
    ......
  }
}
```

14.2.5 从 SSI 调用 Perl 脚本

指令格式如下：

```
<!- # perl sub="module::function" arg="parameter1" arg="parameter2"... >
```

请求对象的方法$r：

$r->args——返回请求参数的方法。

$r->discard_request_body——告诉 Nginx 丢弃请求主体的方法。

$r->filename——返回与 URI 请求想符合的文件名。

$r->has_request_body（function）——如果没有请求的主体，则返回 0。如果请求的主体存在，则建立传递的函数并返回 1。在主体处理的结尾，Nginx 将调用已经建立的处理器。用法示例如代码 14-5：

代码 14-5

```
package hello;

use nginx;

sub handler {
  my $r = shift;

  if ($r->request_method ne "POST") {
    return DECLINED;
  }

  if ($r->has_request_body(\&post)) {
    return OK;
  }

  return 400;
```


www.TopSage.com

```
}

sub post {
  my $r = shift;
  $r->send_http_header;
  $r->print("request_body: \"", $r->request_body, "\"<br/>");
  $r->print("request_body_file: \"", $r->request_body_file, "\"<br/>\n");
  return OK;
}

1;

__END__
```

$r->header_in（header）——检索一个 HTTP 请求头。

$r->header_only——如果只须要返回一个应答头，则为真。

$r->header_out（header, value）——设置一个应答头。

$r->internal_redirect（uri）——使用内部重定向到指定的 URI。重定向只能在 Perl 脚本执行结束后发生。

$r->print（args, ...）——发送数据到客户端。

$r->request_body——当主体没有记录进临时文件时，返回客户端请求主体。因此，为了保证客户端请求主体保留在内存中，须要使用 client_max_body_size 指令限制其大小，并且使用 client_body_buffer_size 指令指定足够大的缓冲区。

$r->request_body_file——返回存储客户端需求主体的文件名。该文件在处理完成后必须被删除。如果需要请求主体总是写入该文件，你须要设置 client_body_in_file_only 指令为 on。

$r->request_method——返回 HTTP 请求方法。

$r->remote_addr——返回客户端 IP 地址。

$r->rflush——马上发送数据到客户端。

$r->sendfile（file [, displacement [, length]）——传输文件的内容给客户端显示。可选的参数用来设置初始位置点及传送数据的长度。严格来说，数据传输只能发生在 Perl 脚本执行完毕后。

$r->send_http_header（type）——添加 Header 到应答中。参数选项“type”用来填写 Header 行“Content-Type”的值。

$r->sleep（milliseconds, handler）——设置在给定请求时间内的指定处理程序，或者停止处理。在这段时间内，Nginx 将继续处理其他请求。在规定的超时时间期满后，Nginx 将运行安装处理程序。请注意，你须要传递一个标识符给函数处理程序。在处理器之间传输数据，你应该使

www.TopSage.com

大家网 TopSage.com

用$r->variable()。用法示例如代码 14-6：

代码 14-6

```
package hello;

use nginx;

sub handler {
   my $r = shift;

  $r->discard_request_body;
  $r->variable("var", "OK");
  $r->sleep(1000, \&next);

  return OK;
}

sub next {
  my $r = shift;

  $r->send_http_header;
  $r->print($r->variable("var"));

  return OK;
}

1;
__END__
```

$r->status（code）——设置 HTTP 应答头。

$r->unescape（text）——使用 unescape 方法对以%XX 十六进制形式编码的文本进程解码。

$r->uri——返回请求的 URI。

$r->variable（name[, value]）——返回或设置指定变量的值。对每个查询来说，变量为局部变量。

14.3 Flv Stream 模块

本模块用于 FLash 播放器以 HTTP 下载方式播放远程 Web 服务器上的 FLV 视频时，支持播放进度条拖动，即支持以 test.flv?start=12345 的方式从指定字节位置下载文件。国内著名的视频分享网站优酷网、土豆网、新浪播客等，都支持这种拖动方式。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_flv_module
```


www.TopSage.com

本模块的配置示例如下：

```
location ~ \.flv$ {
  flv;
}
```

在前面第 11 章第 11.2.1 节中，已详细介绍了“采用 Nginx 的 Flv Stream 模块搭建 HTTP 下载方式的 FLV 视频服务器”，这里不再赘述。

14.3.1 flv 指令

语法：flv

默认值：None

使用环境：location

为当前的 location 开启针对 FLV 视频拖动播放的特殊文件处理。

14.4 HTTP Gzip Static 模块

在一个文件传送给能够接受 Gzip 压缩的客户端浏览器之前，本模块用来检查同名 location 下是否存在以“.gz”结尾的文件。这样做的意图是避免重复压缩相同的文件。

本模块从 Nginx 0.6.24 版本开始被引入，默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_gzip_static_module
```

示例代码如下：

```
gzip_static on;

gzip_http_version   1.1;
gzip_proxied        expired no-cache no-store private auth;
gzip_disable        "MSIE [1-6]\.";
gzip_vary           on;
```

14.4.1 gzip_static 指令

语法：gzip_static on|off

默认值：gzip_static off

使用环境：http, server, location

www.TopSage.com

开启 HTTP Gzip Static 模块。你应该确保压缩文件和未压缩文件的时间戳一样。

14.4.2 gzip_http_version 指令

参见第 13 章第 13.11.5 节的 gzip_http_version 指令。

14.4.3 gzip_proxied 指令

参见第 13 章第 13.11.6 节的 gzip_proxied 指令。

14.5 HTTP Random Index 模块

本模块用于从目录中选择一个随机目录索引。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_random_index_module
```

配置示例代码如下：

```
location / {
 random_index on;
}
```

14.5.1 random_index 指令

语法：random_index [on|off]

默认值：off

使用环境：location

如果在一个指定的 location 中开启本指令，它将为每次访问扫描指定目录内的文件，并发送一个随机选取的文件代替通常的 index.html。以“.”开头的文件不会被选取。

14.6 HTTP Geo IP 模块

本模块基于客户端 IP 地址与 MaxMind()提供的 GeoIP 地址库进行比对，创建一些变量，用来实现地区性负载均衡。本模块适用于 Nginx 0.7.63 和 0.8.6 之后的版本。

www.TopSage.com

大家网 TopSage.com

本模块需要 Geo IP 数据库及读取该数据库的 libgeoip 类库，如代码 14-7 所示。

代码 14-7

```
#下载免费的 geo 城市 IP 数据库
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
#下载免费的 geo 国家 IP 数据库
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/
GeoIP.dat.gz
#下载 libgeoip，在 Debian Linux 操作系统，你可以使用以下方式安装:
sudo apt-get install libgeoip-dev
#在其他 Linux 操作系统，你可以下载源码自己编译安装
wget http://geolite.maxmind.com/download/geoip/api/c/GeoIP.tar.gz
```

在 CentOS Linux 操作系统，你可以使用 yum 命令安装 libgeoip：

```
yum install geoip-devel
```

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_geoip_module
```

配置示例代码如下：

```
http {
  geoip_country  GeoIP.dat;
  geoip_city     GeoLiteCity.dat;
  ...
```

14.6.1 geoip_country 指令

语法：geoip_country path/to/db.dat;

默认值：none

使用环境：http

该指令用于指定判断访问者 IP 地址所属国家的.dat 数据库文件路径。通过该指令的设置，GeoIP 模块会创建以下可用的变量：

$geoip_country_code;——两个字母的国家代码，例如“RU”,“US”。

$geoip_country_code3;——三个字母的国家代码，例如“RUS”,“USA”。

$geoip_country_name;——国家名称，例如：“Russian Federation”,“United States”。

如果你只需要国家名称，则只设置 geoip_country 数据库（1.1MB）即可，因为 geoip_city 数据库（43MB）要大得多，并且所有的数据库将被缓存在内存中。

www.TopSage.com

14.6.2　geoip_city 指令

语法：geoip_city path/to/db.dat;

默认值：none

使用环境：http

该指令用于指定判断访问者 IP 地址所属国家、地区、城市的.dat 数据库文件路径。通过该指令的设置，GeoIP 模块会创建以下可用的变量：

$geoip_city_country_code——两个字母的国家代码，例如“RU”，“US”。

$geoip_city_country_code3——三个字母的国家代码，例如“RUS”，“USA”。

$geoip_city_country_name——国家名称，例如：“Russian Federation”，“United States”（if available）。

$geoip_region——地区名称（省，地区，州，大行政区，联邦政府国土等）例如：“Moscow City”，“DC”（如果存在）。

$geoip_city——城市名称，例如：“Moscow”，“Washington”（如果存在）。

$geoip_postal_code——邮政编码（如果存在）。

$geoip_city_continent_code——洲（指欧、亚、非、南北美、澳、南极洲之一）代码（如果存在）。

$geoip_latitude——纬度（如果存在）。

$geoip_longitude——经度（如果存在）。

14.7　HTTP RealIp 模块

本模块可以修改客户端请求头中的客户端 IP 地址，例如 X-Real-IP 和 X-Forwarded-For。如果 Nginx 位于 Squid 等代理服务器之后，或者位于 F5 Big-IP、NetScaler 等七层负载均衡交换机之后，本模块将非常有用。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_realip_module
```

假设一台 Nginx 位于 Squid 的后端，Squid 设置了 X-Forwarded-For 头信息，记录了用户的真实 IP，但是，如果存在多级代理的情况，X-Forwarded-For 头信息的内容值将变成以逗号分隔的

www.TopSage.com

多个 IP 地址。

```
X-Forwarded-For: 202.108.1.1, 192.168.1.5, 192.168.1.6, 192.168.2.1
```

通过本模块，你可以设置一个可信的代理服务器 IP 列表，Header 头中第一个不受信任的 IP 将作为客户端 IP。例如以上的 X-Forwarded-For 信息中，假设 202.108.1.1 为客户端 IP，其他均为代理服务器 IP，就可以使用以下配置：

```
set_real_ip_from  192.168.1.0/24;
set_real_ip_from  192.168.2.1;
real_ip_header    X-Forwarded-For;
```

这样，X-Forwarded-For 中的 IP 信息就只剩下客户端的 IP 地址 202.108.1.1 了。

14.7.1 set_real_ip_from 指令

语法：set_real_ip_from [the address|CIDR]

默认值：none

使用环境：http, server, location

该指令用于设置可信的代理服务器 IP 地址，这些 IP 地址将在请求转发时被从 Header 头信息中去掉。

14.7.2 real_ip_header 指令

语法：real_ip_header [X-Real-IP|X-Forwarded-For]

默认值：real_ip_header X-Real-IP

使用环境：http, server, location

该指令用于设置转发客户端 IP 地址的 Header 头名称。

14.8 HTTP SSL 模块

本模块用于 HTTPS 支持。

它支持使用以下两个限制检查客户端证书：

（1）不允许指定过期证书的列表。

（2）如果你有一个证书链文件，你无须像 Apache 那样指定每个证书文件。例如金山逍遥网

www.TopSage.com

用户中心使用的中国互联网信息中心（CNNIC）SSL 证书，则是由三个证书链文件构成的。根证书由 Entrust.net 颁发给自己，中级根证书由 Entrust.net 颁发给 CNNIC，域名 my.xoyo.com 的 CRT 证书文件由 CNNIC 颁发，如图 14-1 所示。

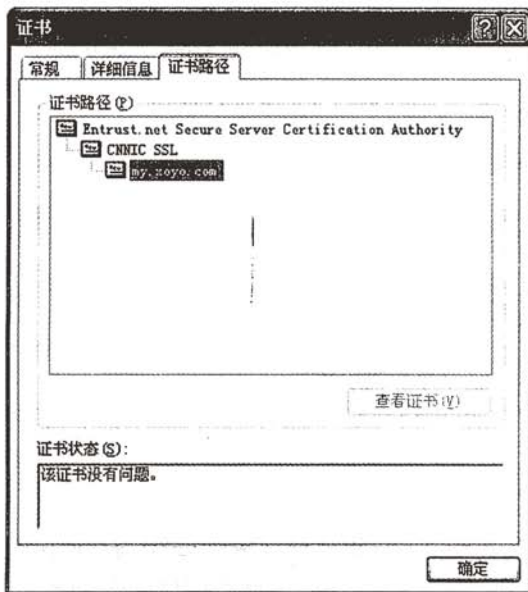


图 14-1 Windows 上查看证书链文件

将三个证书文件的内容（文本格式）按照低级别证书在前的方式，拷贝到一个证书链文件（文本格式）中，即可构成证书链文件，Nginx 配置文件中只须要指定这个证书链文件即可。证书链格式如代码 14-8 所示：

代码 14-8

```
-----BEGIN CERTIFICATE-----
根证书内容
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
中级根证书内容
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
my.xoyo.com 域名证书内容
-----END CERTIFICATE-----
```

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_ssl_module
```

配置示例如代码 14-9：

www.TopSage.com

代码 14-9

```
http {
  server {
    listen              443;
    ssl                 on;
    ssl_certificate     /usr/local/nginx/conf/cert.pem;
    ssl_certificate_key /usr/local/nginx/conf/cert.key;
    keepalive_timeout   70;
  }
}
```

从 Nginx 0.7.14 版本开始，通常习惯在 listen 指令中使用“ssl”参数：

```
server {
  listen 443 ssl;
  ssl_certificate     /usr/local/nginx/conf/cert.pem;
  ssl_certificate_key /usr/local/nginx/conf/cert.key;
  ...
}
```

生成证书：

通常情况下，你可以按照以下步骤生成一个自行颁发的证书，如代码 14-10 所示：

代码 14-10

```
cd /usr/local/nginx/conf
openssl genrsa -des3 -out server.key 1024
openssl req -new -key server.key -out server.csr
cp server.key server.key.org
openssl rsa -in server.key.org -out server.key
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

在 Nginx 配置文件中配置新证书，如代码 14-11：

代码 14-11

```
server {
   server_name YOUR_DOMAINNAME_HERE;
   listen 443;
   ssl on;
   ssl_certificate /usr/local/nginx/conf/server.crt;
   ssl_certificate_key /usr/local/nginx/conf/server.key;
}
```

重启 Nginx，然后访问：

```
https://YOUR_DOMAINNAME_HERE
```

在第 11 章，已介绍了使用 HTTPS（SSL）构建一个安全的 Nginx Web 服务器实例。

www.TopSage.com

14.8.1 在多个 server{......}虚拟主机中使用通配符 SSL 证书

如果你有一个通配符 SSL 证书，例如颁发给*.nginx.org 域名的 SSL 证书，你可以在 http{......}中指定证书文件和私钥文件，那么在各虚拟主机中就可以继承相同的证书。配置示例如代码 14-12：

代码 14-12

```
ssl_certificate     common.crt;
ssl_certificate_key  common.key;

server {
  listen         80;
  server_name     www.nginx.org;
  ...
}

server {
  listen         443 ssl;
  server_name     secure.nginx.org;
  ...
}

server {
  listen         80;
  listen         443;
  server_name     images.nginx.org;
  ...
}
```

14.8.2 ssl 指令

语法：ssl [on|off]

默认值：ssl off

使用环境：main, server

为一个 server{......}虚拟主机开启 HTTPS（SSL）支持。

14.8.3 ssl_certificate 指令

语法：ssl_certificate file

默认值：ssl_certificate cert.pem

www.TopSage.com

大家网 TopSage.com

使用环境：main, server

为当前的虚拟主机指定 PEM 格式的证书文件。一个文件中可以包含多个证书文件（证书链），同样，密钥也必须为 PEM 格式。从 Nginx 0.6.7 版本开始，证书文件的相对路径为 nginx.conf 配置文件所在的目录，而不是 Nginx 安装目录。

14.8.4 ssl_certificate_key 指令

语法：ssl_certificate_key file

默认值：ssl_certificate_key cert.pem

使用环境：main, server

为当前的虚拟主机指定 PEM 格式的私钥文件。从 Nginx 0.6.7 版本开始，证书文件的相对路径为 nginx.conf 配置文件所在的目录，而不是 Nginx 安装目录。

14.8.5 ssl_client_certificate 指令

语法：ssl_client_certificate file

默认值：none

使用环境：main, server

指定 PEM 格式的 CA 证书，用于检查客户端证书。

14.8.6 ssl_dhparam 指令

语法：ssl_dhparam file

默认值：none

使用环境：main, server

指定 PEM 格式的含有 Diffie-Hellman 参数的文件，用于 TLS 会话键。

14.8.7 ssl_ciphers 指令

语法：ssl_ciphers file

默认值：ssl_ciphers ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP

www.TopSage.com

使用环境：main, server

指定许可密码的描述。密码以 OpenSSL 支持的格式指定，例如：

```
ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

使用以下命令可以查看 OpenSSL 支持的完整格式列表：

```
openssl ciphers
```

14.8.8 ssl_crl 指令

语法：ssl_crl file

默认值：none

使用环境：http, server

该指令（Nginx 0.8.7 以上版本开始支持）用于指定一个 PEM 格式的证书吊销文件，用于检查客户端证书。

14.8.9 ssl_prefer_server_ciphers 指令

语法：ssl_prefer_server_ciphers [on|off]

默认值：ssl_prefer_server_ciphers off

使用环境：main, server

对 SSLv3 和 TLSv1 协议的服务器端密码需求优先级高于客户端密码。

14.8.10 ssl_protocols 指令

语法：ssl_protocols [SSLv2] [SSLv3] [TLSv1]

默认值：ssl_protocols SSLv2 SSLv3 TLSv1

使用环境：main, server

该指令用于指定使用的 SSL 协议。

14.8.11 ssl_verify_client 指令

语法：ssl_verify_client on|off|ask

www.TopSage.com

默认值：ssl_verify_client off

使用环境：main, server

该指令用于开启客户端证书验证。参数“ask”在客户端主动提出检查证书时，对客户端证书进行检查。

14.8.12 ssl_verify_depth 指令

语法：ssl_verify_depth number

默认值：ssl_verify_depth 1

使用环境：main, server

设置客户端证书链的深度。

14.8.13 ssl_session_cache 指令

语法：ssl_session_cache off|none|builtin:size and/or shared:name:size

默认值：ssl_session_cache off

使用环境：main, server

该指令设置用来存储 SSL 会话的缓存类型和大小。缓存类型为：

off——硬关闭：Nginx 明确告诉客户端这个会话不可用。

none——软关闭：Nginx 告诉客户端会话能够被重用，但是 Nginx 实际上不会重用它们。这是为某些邮件客户端使用的一种变通方法，可以被用于邮件代理和 HTTP 服务器中。

builtin——OpenSSL 内置缓存，仅可用于一个工作进程。缓存大小用会话数来指定。注意：使用该指令会导致内存碎片，当使用此功能时须要考虑。

shared——位于所有工作进程的共享缓存。缓存的大小用字节数指定，1MB 缓存能够容纳大约 4 000 会话。每个共享缓存必须拥有自己的名称。同名的缓存可以用于多个虚拟主机。

你也可以同时使用 builtin 和 shared，示例如下：

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

然而，只使用共享内存而不使用 builtin 缓存，将更有效。

www.TopSage.com

大家网 TopSage.com

14.8.14 ssl_session_timeout 指令

语法：ssl_session_timeout time

默认值：ssl_session_timeout 5m

使用环境：main, server

设置客户端能够重复使用存储在缓存中的会话参数时间。

该模块支持一些非标准的错误代码，可以借助 error_page 指令来做 debug 调试：

495——检查客户端证书时发生错误。

496——客户端不允许必须的证书。

497——正常的请求发送到 HTTPS。

在调试完成后，可以取得一些变量，例如 $request_uri、$uri、$arg 等。Nginx 的 Http SSL 模块支持这些内置变量：

$ssl_cipher：返回既定 SSL 连接中使用的密码行。

$ssl_client_serial：返回既定 SSL 连接的客户端证书的序列号。

$ssl_client_s_dn：返回既定 SSL 连接的客户端证书的 DN 主题行。

$ssl_client_i_dn：返回既定 SSL 连接的客户端证书的 DN 发行站行。

$ssl_protocol：返回既定 SSL 连接的协议。

$ssl_session_id：在 Nginx 0.8.20 以上版本支持该变量。

$ssl_client_cert

$ssl_client_raw_cert

$ssl_verify "SUCCESS"：如果客户端证书验证通过，该变量的值为“SUCCESS”。

14.8.15 ssl_engine 指令

语法：ssl_engine

该指令指定允许去使用的 OpenSSL 引擎，例如 Padlock。须要安装一个近期发布的 OpenSSL 版本。

www.TopSage.com

14.9 HTTP Stub Status 模块

本模块主要用于查看 Nginx 的一些状态信息。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_stub_status_module
```

配置示例如代码 14-13：

代码 14-13

```
server
{
   listen  80;
   server_name  status.yourdomain.com;

   location / {
      stub_status on;
      access_log  off;
      allow 192.168.1.2;
      deny all;
   }
}
```

14.9.1 stub_status 指令

语法：stub_status on

默认值：None

使用环境：location

该指令用于开启 Nginx 状态信息。

访问以上示例中配置的 *http://status.yourdomain.com/*，显示的 Nginx 状态信息示例如下：

```
Active connections: 17580
server accepts handled requests
 38810620 38810620 298655730
Reading: 68 Writing: 1219 Waiting: 16293
```

Active connections——对后端发起的活动连接数。

Server accepts handled requests——Nginx 总共处理了 43 629 083 个连接，成功创建 43 629 083 次握手（证明中间没有失败的），总共处理了 259 552 136 个请求。

Reading——Nginx 读取到客户端的 Header 信息数。

Writing——Nginx 返回给客户端的 Header 信息数。

Waiting——开启 keep-alive 的情况下，这个值等于 active -（reading + writing），意思就是 Nginx 已经处理完成，正在等候下一次请求指令的驻留连接。

所以，在访问效率高，请求很快被处理完毕的情况下，Waiting 数比较多是正常的。如果 reading + writing 数较多，则说明并发访问量非常大，正在处理过程中。

14.10 HTTP Sub 模块

本模块主要用来搜索并替换 Nginx 应答内容中的文本。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_sub_module
```

示例如下：

```
location / {
  sub_filter     </head>
  '</head><script language="javascript" src="$script"></script>';
  sub_filter_once on;
}
```

14.10.1 sub_filter 指令

语法：sub_filter text substitution

默认值：none

使用环境：http, server, location

该指令允许使用一些其他的文本替换 Nginx 应答内容中的一些文本。匹配不区分大小写，替换的文本可以包含变量。每个 location 只能指定一个替换规则。

14.10.2 sub_filter_once 指令

语法：sub_filter_once on|off

默认值：sub_filter_once on

使用环境：http, server, location

www.TopSage.com



该指令的值设置为 off 时，允许搜索并替换所有匹配行。该指令的默认值为 on，只替换第一个匹配项。

14.10.3 sub_filter_types 指令

语法：sub_filter_types mime-type [mime-type ...]

默认值：sub_filter_types text/html

使用环境：http, server, location

该指令用于指定 sub_filter 指令将检查哪一内容类型，默认值为 text/html。

14.11 HTTP Dav 模块

本模块用来设置允许 HTTP 和 WebDAV 方法：PUT、DELETE、MKCOL、COPY、MOVE。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_dav_module
```

示例如代码 14-14：

代码 14-14

```
location / {
 root     /data/www;
 client_body_temp_path  /data/client_temp;

 dav_methods  PUT DELETE MKCOL COPY MOVE;

 create_full_put_path   on;
 dav_access             group:rw  all:r;

 limit_except  GET {
  allow  192.168.1.0/32;
  deny   all;
 }
}
```

14.11.1 dav_access 指令

语法：dav_access user:permissions [users:permissions] ...

默认值：dav_access user:rw

www.TopSage.com

使用环境：http, server, location

该指令用于指定文件和目录的访问权限，示例如下：

```
dav_access user:rw group:rw all:r;
```

如果指定任何允许的 groups 或 all，则无须为 user 指定权限：

```
dav_access group:rw all:r;
```

14.11.2 dav_methods 指令

语法：dav_methods [off|put|delete|mkcol|copy|move] ...

默认值：dav_methods off

使用环境：http, server, location

该指令用来设置允许指定的 HTTP 和 WebDAV 方法。如果设置为 off，所有的方法将被禁止，并且忽略剩余的参数。

PUT 方法的目的文件必须在相同分区的临时文件存储目录中存在，使用 client_body_temp_path 指令在 section 区域中设置。

当一个文件被使用 PUT 方法创建时，将使用 Date 头信息作为该文件的修改时间。

14.11.3 create_full_put_path 指令

语法：create_full_put_path on|off

默认值：create_full_put_path off

使用环境：http, server, location

默认情况（off）下，PUT 方法只能在已经存在的目录中创建文件。该指令允许 Nginx 创建目录。

14.12 Google Perftools 模块

本模块允许使用 Google 公司开发的性能优化工具 Google Performance Tools（*http://code.google.com/p/google-perftools/*）。该模块可在 Nginx 0.6.29 之后的版本中使用。

Google Performance Tools 的安装步骤如代码 14-15 所示：

www.TopSage.com

代码 14-15

```
#64 位操作系统须要先安装 libunwind
wget http://ftp.twaren.net/Unix/NonGNU/libunwind/libunwind-0.99.tar.gz
tar zxvf libunwind-0.99.tar.gz
cd libunwind-0.99/
CFLAGS=-fPIC ./configure
make CFLAGS=-fPIC
make CFLAGS=-fPIC install
cd ../

wget http://google-perftools.googlecode.com/files/google-perftools-1.4.tar.gz
tar zxvf google-perftools-1.4.tar.gz
cd google-perftools-1.4/
./configure --prefix=/usr
make && make install
cd ../
/sbin/ldconfig
```

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-google_perftools_module
```

示例如下：

```
google_perftools_profiles /path/to/profile;
```

Profile 将以/path/to/profile.<worker_pid>格式存储。

14.12.1 google_perftools_profiles 指令

语法：google_perftools_profiles path

默认值：none

该指令用于指定 profiles 基础文件名，工作进程的 PID 将被添加到指定的文件中。

14.13 HTTP XSLT 模块

XSLT 是一种用于将 XML 文档转换为 XHTML 文档或其他 XML 文档的语言。本模块是通过一个或多个 XSLT 模板转换 XML 应答内容的过滤器。本模块适用于 Nginx 0.7.8 以后版本。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_xslt_module
```

配置示例如下：

www.TopSage.com




```
location / {
 xml_entities       /site/dtd/entities.dtd;
 xslt_stylesheet    /site/xslt/one.xslt   param=value;
 xslt_stylesheet    /site/xslt/two.xslt;
}
```

14.13.1 xslt_entities 指令

语法：xml_entities <path>

默认值：no

使用环境：http, server, location

用于指定 DTD 描述文件（XML 实体）。DTD 实际上可以看作一个或多个 XML 文件的模板，这些 XML 文件中的元素、元素的属性、元素的排列方式/顺序、元素能够包含的内容等，都必须符合 DTD 中的定义。此文件在配置阶段被编译。由于技术原因，不能对正在处理的 XML 文件指定实体，但是会用一个特别指定的文件来代替。在这个文件中，没有必要来描述处理 XML 的结构，只须说明必须的符号元素即可，例如：

```
<! ENTITY of nbsp "  ">
```

14.13.2 xslt_stylesheet 指令

语法：xslt_stylesheet template [parameter[[parameter...]]

默认值：no

使用环境：http, server, location

用于指定 XSLT 模板的参数。模板在配置阶段被编译。参数按照以下方法指定：

```
 param=value
```

你可以为每行指定任何参数，或者使用“:”分割多个参数。如果参数自身包含字符“:”，请使用“%3A”转义。此外，如果参数包含非字母数字的字符，libxslt 要求字符串参数应该用单引号或双引号引用。示例如下：

```
 param1='http%3A//www.example.com': param2=value2
```

它可以使用变量作为参数，例如，参数值可以用一个变量来取代：

```
 location / {
  xslt_stylesheet /site/xslt/one.xslt
  $arg_xslt_params
```


www.TopSage.com

```
  param1='$value1': param2=value2
  param3=value3;
}
```

可以指定多个模板，在这种情况下，将按照它们的声明顺序链接在一起。

14.13.3 xslt_types 指令

语法：xslt_types mime-type [mime-type...]

默认值：xslt_types text/xml

使用环境：http, server, location

允许处理除了“text/xml”之外的指定 MIME 类型。如果 XSLT 输出模式为 HTML，应答的 MIME 类型可以更改为“text/HTML”。

14.14 HTTP Secure Link 模块

本模块计算和请求需要安全性令牌的URL地址。在Nginx 0.7.18以上版本中可以使用本模块。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_secure_link_module
```

配置示例代码如下：

```
location /prefix/ {
  secure_link_secret  secret_word;

  if ($secure_link = "") {
    return 403;
  }
}
```

14.14.1 secure_link_secret 指令

语法：secure_link_secret secret_word

默认值：none

使用环境：location

该指令用于指定一个密码去校验请求。一个被保护链接的完整 URL 如下：

```
/prefix/hash/reference
```


www.TopSage.com

该指令将按照以下方式进行哈希计算：

```
md5 (reference, secret_word);
```

prefix 为 location 区块的位置范围，不能为“/”。secure_link 只能用于非 root 路径。

配置示例代码如 14-16：

代码 14-16

```
server
{
   listen      80;
   server_name  www.yourdomain.com;
   index index.html index.htm;
   root  /data0/htdocs/www;

   location /download {
       secure_link_secret password;
       if ($secure_link = "") {
          return 403;
       }
       rewrite (.*) /download/$secure_link  break;
   }
}
```

假设有两个文件/data0/htdocs/www/download/demo.mp4 和/data0/htdocs/www/download/video/demo.mp4，可通过 PHP 程序生成一个 md5 串：

```
<?php
echo md5("demo.mp4"."password")."\n";
echo md5("video/demo.mp4"."password")."\n";
?>
```

执行该 PHP 程序得出：

```
626e08fd9d6822090b3d684a8ef325d7
ceab52da294cb1a77671d315000bf6ef
```

通过以下 URL 访问 Nginx，就可以下载这两个文件：

```
http://www.yourdomain.com/download/626e08fd9d6822090b3d684a8ef325d7/demo.mp4
http://www.yourdomain.com/download/626e08fd9d6822090b3d684a8ef325d7/video/demo.mp
```

14.14.2 $secure_link 变量

自动设置 URL 的基准部分，并与 prefix 和 hash 分开。如果 hash 错误，变量$secure_link 的值将为空。

www.TopSage.com

大家网 TopSage.com

14.15 HTTP Image Filter 模块

本模块是一个转换 JPEG、GIF、PNG 图片的过滤器。

本模块默认是不会编译进 Nginx 的，如果你要使用该模块，则要在编译安装 Nginx 时指定：

```
./configure --with-http_image_filter_module
```

本模块需要操作系统安装 libgd 类库。

配置示例如代码 14-17：

代码 14-17

```
location /img/ {
   proxy_pass     http://backend;
   image_filter   resize  150 100;
   error_page     415    = /empty;
}

location = /empty {
   empty_gif;
}
```

14.15.1 image_filter 指令

语法：image_filter（test|size|resize width height|crop width height）

默认值：none

使用环境：location

它指定适用于图片的转换类型。

test: 检查应答是否确实为一个图片格式 JPEG、GIF、PNG，如果不是，返回 415。

size: 以 JSON 格式给出图片的信息，示例如下：

```
{ "img" : { "width": 100, "height": 100, "type": "gif" } }
```

或者当一个错误发生时，返回：

```
{}
```

resize: 按比例缩小图片到指定的大小。

crop: 按比例缩小图片到指定的大小，并裁减掉超过指定大小的图片区域。

www.TopSage.com

14.15.2 image_filter_buffer 指令

语法：image_filter_buffer size

默认值：1M

使用环境：http, server, location

设置读取图片的最大值。

14.15.3 image_filter_jpeg_quality 指令

语法：image_filter_jpeg_quality [0...100]

默认值：75

使用环境：http, server, location

设置处理 JPEG 图片的质量损耗比率。最大推荐者为 95。

14.15.4 image_filter_transparency 指令

语法：image_filter_transparency on|off

默认值：on

使用环境：http, server, location

该指令允许你在 GIF 和基于调色板的 PNG 禁止使用图片透明，从而提高图片质量。

不管这个指令如何设置，真彩色 PNG alpha-channels 总是存在。

注意：灰度 PNG 未经测验，但是能够被当成真彩色来处理。

www.TopSage.com

第15章 Nginx的邮件模块

Nginx 除了可用作 HTTP 服务器外，还可用作邮件代理，支持 IMAP、POP3、SMTP 协议。

IMAP（Internet Message Access Potocol）是一种用于邮箱访问的协议，使用 IMAP 协议可以在客户端管理服务器上的邮箱。它与 POP 不同，邮件是保留在服务器上而不是下载到本地，在这一点上 IMAP 是与 Webmail 相似的。但 IMAP 有比 Webmail 更好的地方，它比 Webmail 更高效和安全，可以离线阅读等。如果想用 Outlook Express、Foxmail 试试，只要配置好一个账号，将邮件接收服务器设置为 IMAP 服务器就可。此外，IMAP 还提供了一个更好的方法供您从多个设备中访问邮件。不管您是在工作中检查邮件，或者通过手机检查邮件，还是在家里检查邮件，IMAP 都能确保您从任何设备随时访问新邮件。总之，IMAP 可从整体上为用户带来更为可靠的体验。虽然 POP 更易丢失邮件或多次下载相同的邮件，但 IMAP 通过邮件客户端与邮件服务器之间的双向同步功能很好地避免了这些问题。例如，Gmail、QQ 邮箱都支持 IMAP 协议访问。

POP3（Post Office Protocol 3）即邮局协议的第 3 个版本，它规定怎样将个人计算机连接到 Internet 的邮件服务器和下载电子邮件的电子协议。它是 Internet 电子邮件的第一个离线协议标准，POP3 允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上，同时删除保存在邮件服务器上的邮件，而 POP3 服务器则是遵循 POP3 协议的接收邮件服务器，用来接收电子邮件的。

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议，它是一组用于由源地址到目的

www.TopSage.com

地址传送邮件的规则，由它来控制信件的中转方式。SMTP 协议属于 TCP / IP 协议族，它帮助每台计算机在发送或中转信件时找到下一个目的地。通过 SMTP 协议所指定的服务器，可以把 E-mail 寄到收信人的服务器上，整个过程只要几分钟。SMTP 服务器则是遵循 SMTP 协议的发送邮件服务器，用来发送或中转你发出的电子邮件。

Nginx 使用外部的类 HTTP 服务器去获知应该连接到哪个 IMAP/POP 后端服务器。

Nginx 在 HTTP Header 头中传递认证信息如下：

```
GET /auth HTTP/1.0
Host: auth.server.hostname
Auth-Method: plain
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.168.1.1
```

正常的应答为：

```
HTTP/1.0 200 OK       # 此行实际上被忽略，可能根本不存在
Auth-Status: OK
Auth-Server: 192.168.1.10
Auth-Port: 110
Auth-User: newname   # 你可以忽略登录到后端的用户名
```

在 POP3 中，为了避免发送明文口令的问题，有一种新的认证方法，命令为 APOP，使用 APOP，口令在传输之前被加密。当 POP3 认证使用 APOP 时，你必须返回 Auth-Pass：

```
HTTP/1.0 200 OK       # 此行实际上被忽略，可能根本不存在
Auth-Status: OK
Auth-Server: 192.168.1.10
Auth-Port: 110
Auth-User: newname   # 你可以忽略登录到后端的用户名
Auth-Pass: password  # 这必须用户的明文密码
```

认证失败的应答结果为：

```
HTTP/1.0 200 OK       # 此行实际上被忽略，可能根本不存在
Auth-Status: Invalid login or password
Auth-Wait: 3         # Nginx 在重新读取客户端登录账号、密码之前会等待 3 秒钟
# client's login/passwd again
```

在本章的指令介绍中，指令的“使用环境”是该指令在 Nginx 配置文件中使用的位置，例如使用环境为“mail, server”，表示该指令可以在以下位置使用：

mail { ...... }大括号内；server { ...... }大括号内。

www.TopSage.com

15.1 Nginx 邮件核心模块

15.1.1 auth 指令

从 Nginx 0.5.15 版本开始，该指令重命名为 pop3_auth。

15.1.2 imap_capabilities 指令

语法：imap_capabilities “capability1” [“capability2” .. “capabilityN”]

默认值：“IMAP4” “IMAP4rev1” “UIDPLUS”

使用环境：main, server

在客户端发送 IMAP 命令 CAPABILITY 的时候，通过这个指令，你可以设置 IMAP 协议扩展列表。如果启用 starttls 指令，STARTTLS 将被自动添加。

目前，标准的 IMAP 扩展发布在网站 www.iana.org。imap_capabilities 设置示例如下：

```
mail {
  imap_capabilities NAMESPACE SORT QUOTA;
}
```

15.1.3 imap_client_buffer 指令

语法：iimap_client_buffer size

默认值：4k/8k

使用环境：main, server

使用本指令可以设置 IMAP 命令的读取缓冲区。默认值等于页大小（该值取决于操作系统，为 4k 或 8k）。

15.1.4 listen 指令

语法：listen address:port [bind]

默认值：no

使用环境：server

www.TopSage.com

该指令用于指定接收请求的服务器地址和端口。它也可以单一指定服务器地址或端口，此外，地址客户使用服务器名称，示例如下：

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;
```

IPv6 地址（Nginx 0.7.58 以上版本支持）请在中括号中设置，示例如下：

```
listen [::]:8000;
listen [fe80::1];
```

在 listen 指令中，可以设置系统调用 bind（2）。

15.1.5 pop3_auth 指令

语法：pop3_auth [plain] [apop] [cram-md5]

默认值：plain

使用环境：main, server

该指令可以设置 POP3 客户端认证的许可方法：

plain——USER/PASS , AUTH PLAIN , AUTH LOGIN

apop——APOP

cram-md5——AUTH CRAM-MD5

15.1.6 pop3_capabilities 指令

语法：pop3_capabilities “capability1” [“capability2” .. “capabilityN”]

默认值：“TOP” “USER” “UIDL”

使用环境：main, server

在客户端发送 POP3 命令 CAPA 的时候，通过这个指令，你可以设置 POP3 协议扩展列表。如果你启用 starttls 指令，STLS 将被自动添加。SASL 将在 auth 指令中添加。

15.1.7 protocol 指令

语法：protocol [pop3 | imap | smtp] ;

www.TopSage.com

大家网 TopSage.com

默认值：IMAP

使用环境：server

该指令用于设置虚拟主机 server{...}块使用的协议。

15.1.8 server 指令

语法：server {...}

默认值：no

使用环境：mail

该指令用于配置虚拟主机。这里没有明确区分基于 IP 地址的虚拟主机和基于域名的虚拟主机（客户端请求中 Header 头 Host 行的值）。该指令用于配置虚拟主机。在 server {......}中，使用 listen 指令来为一个虚拟主机设置监听的 IP 和端口，使用 server_name 指令来设置不同的虚拟主机名称。

15.1.9 server_name 指令

语法：server_name name fqdn_server_host

默认值：The name of the host, obtained through gethostname()

使用环境：mail, server

该指令用于设置虚拟主机的主机名，示例如下：

```
server {
 server_name   example.com  www.example.com;
}
```

server_name 参数中的第一个名称将成为服务器的基础名称。默认名称为被使用的服务器名称（hostname），你可以使用通配符“*”代替名称的首部分：

```
server {
 server_name   example.com  *.example.com;
}
```

上述示例中的两个名称可以合并成以下的一个：

```
server {
 server_name  .example.com;
}
```

如果在客户端请求中没有 Header 头“Host”，或者 Header 头不匹配任何 server_name，服务

www.TopSage.com

器的基础名称将被用于 HTTP 重定向。你可以使用“*”强制 Nginx 在 HTTP 重定向中使用 Host 头（注意：“*”不能作为虚拟主机的第一个名称，但是，你可以使用一个伪名称“_”来代替第一个名称）：

```
server {
  server_name example.com *;
}
server {
  server_name _ *;
}
```

15.1.10 smtp_auth 指令

语法：smtp_auth [login] [plain] [cram-md5] ;

默认值：login plain

使用环境：main, server

该指令可以设置 SMTP 客户端认证的许可方法：

login——AUTH LOGIN

plain——AUTH PLAIN

cram-md5——AUTH CRAM-MD5

15.1.11 smtp_capabilities 指令

语法：smtp_capabilities “capability1” [“capability2” .. “capabilityN”]

默认值：no

使用环境：main, server

在客户端发送 SMTP 命令 EHLO 的时候，通过这个指令，你可以设置 SMTP 协议扩展列表。此列表通过 smtp_auth 指令启用的方法自动扩展。目前，标准的 SMTP 扩展发布在网站 www.iana.org。

15.1.12 so_keepalive 指令

语法：so_keepalive on|off;

默认值：off

www.TopSage.com

使用环境：main, server

通过该指令，你可以为连接到 IMAP/POP3 后端服务器的套接字设置 SO_KEEPALIVE 选项。在 FreeBSD 中，keepalive 选项将被用于所有的连接，它可以通过内核参数（见 sysctl net.inet.tcp.always_keepalive）关闭。

15.1.13 timeout 指令

语法：timeout milliseconds;

默认值：60000

使用环境：main, server

通过该指令，你可以设置代理服务器连接到后端的超时时间。

15.2 Nginx 邮件认证模块

Nginx 邮件认证模块示例如下：

```
{
 auth_http          localhost:9000/cgi-bin/nginxauth.cgi;
 auth_http_timeout   5;
}
```

15.2.1 auth_http 指令

语法：auth_http URL

默认值：no

使用环境：mail, server

通过该指令，你可以为认证设置 URL 到扩展的类 HTTP 服务器。

15.2.2 auth_http_header 指令

语法：auth_http_header header value

默认值：no

使用环境：mail, server

www.TopSage.com

大家网
TopSage.com

通过该指令，你可以在身份认证的过程中添加一个 HTTP 头和它的值。这使得它可以使用一个共享的秘密字符串，确保请求是由 Nginx 应答。

示例如下：

```
auth_http_header X-NGX-Auth-Key "secret_string";
```

15.2.3 auth_http_timeout 指令

语法：auth_http_timeout milliseconds;

默认值：60000

使用环境：mail, server

通过该指令，你可以设置认证过程的超时时间。

15.3 Nginx 邮件代理模块

Nginx 可以代理 IMAP、POP3、SMTP 协议。

15.3.1 proxy 指令

语法：proxy on | off

默认值：off

使用环境：mail, server

通过该指令，你可以开启或关闭邮件代理。

15.3.2 proxy_buffer 指令

语法：proxy_buffer size

默认值：4k/8k

使用环境：mail, server

通过该指令，你可以设置代理连接的缓冲区大小。默认值等于页大小（该值取决于操作系统，为 4k 或 8k）。

www.TopSage.com

15.3.3 proxy_pass_error_message 指令

语法：proxy_pass_error_message on | off

默认值：off

使用环境：mail, server

通过该指令，你可以传递从后端被代理服务器获取的错误信息给客户端。通常情况下，如果成功通过 Nginx 的认证，后端被代理服务器将不会传递错误信息给客户端。

但是在一些正确密码的应答过程中，POP3 服务器错误属于正常的行为。例如 CommuniGatePro 通知用户关于邮箱存储空间满（或者其他事件）将在认证时周期性地发出错误信息。在这种情况下，有必要设置 proxy_pass_error_message 指令的参数为 on。

15.3.4 proxy_timeout 指令

语法：proxy_timeout time

默认值：24h

使用环境：mail, server

通过该指令，你可以设置代理连接的超时时间。

15.3.5 xclient 指令

语法：xclient on | off

默认值：on

使用环境：mail, server

通过该指令，你可以允许或禁止连接 SMTP 后端服务器时使用 XCLIENT 命令。这将允许后端强制限制客户端连接基于 IP/HELO/LOGIN。

如果 xclient 指令的参数为 on，Nginx 将首先传递以下信息给后端服务器：

```
EHLO server_name
```

然后：

```
XCLIENT PROTO=ESMTP HELO=client_helo ADDR=client_ip LOGIN=authentificated_user
NAME=[UNAVAILABLE]
```


www.TopSage.com

15.4 Nginx 邮件 SSL 模块

本模块为 POP3/IMAP/SMTP 提供 SSL/TLS 支持。配置几乎与 HTTP SSL 模块相同，但是不支持检查客户端证书。

15.4.1 ssl 模块

语法：ssl on | off

默认值：ssl off

使用环境：mail, server

为所在虚拟主机启用或禁用 SSL/TLS。

15.4.2 ssl_certificate 指令

语法：ssl_certificate file

默认值：cert.pem

使用环境：mail, server

为所在虚拟主机指定 PEM 格式的证书文件。相同的文件客户包含其他的证书，并且密钥也是 PEM 格式。

15.4.3 ssl_certificate_key 指令

语法：ssl_certificate_key file

默认值：cert.pem

使用环境：mail, server

为所在虚拟主机指定 PEM 格式的密钥文件。

15.4.4 ssl_ciphers 指令

语法：ssl_ciphers file ciphers

www.TopSage.com



默认值：ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP

使用环境：mail, server

指出允许的密码格式描述，密码被指定为 OpenSSL 支持的格式。

15.4.5 ssl_prefer_server_ciphers 指令

语法：ssl_prefer_server_ciphers on | off

默认值：off

使用环境：mail, server

要求 SSLv3 和 TLSv1 协议的服务器密码优先于客户端的密码。

15.4.6 ssl_protocols 指令

语法：ssl_protocols [SSLv2] [SSLv3] [TLSv1]

默认值：SSLv2 SSLv3 TLSv1

使用环境：mail, server

该指令用于指定使用的 SSL 协议。

15.4.7 ssl_session_cache 指令

语法：ssl_session_cache [builtin[:size [shared:name:size]

默认值：builtin:20480

使用环境：mail, server

该指令用于设置存储 SSL 会话的缓存类型和大小。缓存类型如下：

builtin——OpenSSL 内建缓存，只能在一个工作进程中使用。缓存大小为会话数。

shared——缓存在所有工作进程中共享。缓存的大小以字节数指定，1MB 的缓存客户容纳大约 4 000 个会话。每个共享缓存必须有独自的名称，同名的缓存客户在不同的虚拟主机中使用。

可以同时使用两种缓存类型，例如：

```
ssl_session_cache  builtin:1000  shared:SSL:10m;
```

然而，只使用共享缓存而不使用内建缓存要更有效。

www.TopSage.com

15.4.8 ssl_session_timeout 指令

语法：ssl_session_timeout time

默认值：5m

使用环境：mail, server

该指令用于设置客户端可以重复使用存储在缓存中的会话参数时间。

15.4.9 starttls 指令

语法：starttls on | off | only

默认值：off

使用环境：mail, server

on——允许使用 POP3 的 STLS 命令和 IMAP/SMTP 的 STARTTLS 命令。

off——不允许使用命令 STLS/STARTTLS。

only——开启 STLS/STARTTLS 支持，并且需要客户端使用 TLS 编码。

15.5 Nginx 邮件模块配置实例

在这一节中，我们通过使用 PHP 脚本来进行后端认证，以展示如何配置 Nginx 邮件模块：

（1）假设你的 POP3/IMAP 邮件代理服务器 Nginx 运行在 192.168.1.1。

（2）假设你有两台 POP3/IMAP 后端邮件服务器：192.168.22 和 192.168.33。

（3）假设你有一台认证和重定向服务器：192.168.1.44。

（4）认证脚本为：/mail/auth.php。

192.168.1.1 服务器上的 nginx.conf 配置文件如代码 15-1 所示：

代码 15-1

```
user  nobody;
worker_processes  1;
error_log  logs/error.log  info;
pid        logs/nginx.pid;
```


www.TopSage.com

```
events {
  worker_connections  1024;
  multi_accept on;
}

mail {
  auth_http  192.168.1.44:80/mail/auth.php;
  pop3_capabilities  "TOP"  "USER";
  imap_capabilities  "IMAP4rev1"  "UIDPLUS";

  server {
    listen     110;
    protocol   pop3;
    proxy     on;
  }

  server {
    listen     143;
    protocol   imap;
    proxy     on;
  }
}
```

192.168.1.44 服务器上的/mail/auth.php 脚本内容如代码 15-2 所示：

代码 15-2

```
<?php
/*
Nginx sends headers as
Auth-User: somuser
Auth-Pass: somepass
On my php app server these are seen as
HTTP_AUTH_USER and HTTP_AUTH_PASS
*/
if (!isset($_SERVER["HTTP_AUTH_USER"] ) || !isset($_SERVER["HTTP_AUTH_PASS"] )){
  fail();
}
$username=$_SERVER["HTTP_AUTH_USER"] ;
$userpass=$_SERVER["HTTP_AUTH_PASS"] ;
$protocol=$_SERVER["HTTP_AUTH_PROTOCOL"] ;
// default backend port
$backend_port=110;
if ($protocol=="imap") {
  $backend_port=143;
}
if ($protocol=="smtp") {
  $backend_port=25;
}
// nginx likes ip address so if your
// application gives back hostname, convert it to ip address here
$backend_ip["mailhost01"] ="192.168.1.22";
```


www.TopSage.com




```
$backend_ip["mailhost02"] ="192.168.1.33";
// Authenticate the user or fail
if (!authuser($username,$userpass){
  fail();
  exit;
}
// Get the server for this user if we have reached so far
$userserver=getmailserver($username);
// Get the ip address of the server
// We are assuming that you backend returns hostname
// We try to get the ip else return what we got back
$server_ip=(isset($backend_ip[$userserver] )?$backend_ip[$userserver] :$userserv
er;
// Pass!
pass($server_ip, $backend_port);

//END


function authuser($user,$pass){
  // put your logic here to authen the user to any backend
  // you want (datbase, ldap, etc)
  // for example, we will just return true;
  return true;
}

function getmailserver($user){
  // put the logic here to get the mailserver
  // backend for the user. You can get this from
  // some database or ldap etc
  // dummy logic, all users that start with a,c,f and g get mailhost01
  // the others get mailhost02
  if in_array(substr($user,0,1), array("a", "c", "f", "g")){
    return "mailhost01";
  } else {
    return "mailhost02";
  }
}

function fail(){
  header("Auth-Status: Invalid login or password");
  exit;
}

function pass($server,$port){
  header("Auth-Status: OK");
  header("Auth-Server: $server");
  header("Auth-Port: $port");
  exit;
}
?>
```


www.TopSage.com

Broadview
技术凝聚实力・专业创新出版

实战Nginx：

取代Apache的高性能Web服务器



Nginx是俄罗斯人Igor Sysoev编写的一款高性能的HTTP和反向代理服务器。Nginx选择了epoll和kqueue作为网络I/O模型，在高连接并发的情况下，Nginx是Apache服务器不错的替代品，它能够支持高达50 000个并发连接数的响应，运行稳定，且内存、CPU等系统资源消耗非常低。

本书主要分为4个部分：

- 第1部分为基础篇，介绍了Nginx服务器的安装与配置方法；
- 第2部分为进阶篇，重点介绍了Nginx的配置优化方法、Nginx与PHP/Ruby/Python/JSP/Perl/Memcached的结合配置方法、Nginx HTTP反向代理与负载均衡的配置与优化、Nginx模块开发等，最后还分析了新浪的开源软件项目——基于Nginx的NCACHE网页缓存系统；
- 第3部分为实战篇，分析了Nginx在国内知名网站（如新浪播客、搜狐博客等）中的应用案例；
- 第4部分为模块篇，对Nginx的基本模块和第三方模块进行了集中介绍。

本书是为对配置管理Nginx服务器感兴趣的读者准备的，适用于以前没有接触过Nginx，或者对Nginx有一些了解并希望能够进一步深入学习的专业系统工程师、个人网站站长及Linux/Unix从业人员。

2008年年底张宴加入了金山逍遥网，作为逍遥网技术部平台组组长，张宴带领组员重新设计了逍遥网的系统架构并加以实施。新系统架构用Nginx替换了原来所有的Apache服务器，既减少了服务器数量和带宽，又提高了整个平台的性能及易维护性。逍遥网全新的系统架构，充分展现了Nginx处理高并发的能力和易维护等特性。

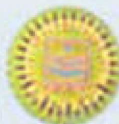
——张鹏翼，金山逍遥网技术部经理

图书分类 Web开发


ISBN 978-7-121-10247-9
9 787121 102479 >



策划编辑：徐定翔
责任编辑：周　筠
责任美编：杨小勤



本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

定价：55.00元